CENA 10.000 zł

ISSN 0867-3918 INDEKS 377112

64 PLUS 4 & AMIGA

5/'92

MIESIĘCZNIK UŻYTKOWNIKÓW KOMPUTERÓW COMMODORE



UWAGA! Bydgoska firma SYSTEM opracowała serię polskich programów edukacyjnych (ortografia, historia, geografia, fizyka i chemia) na cartridge'ach przeznaczonych dla C-64.

Podany w przystępnej formie duży zasób materiału umożliwia szybkie i skuteczne przyswojenie wiadomości.

Bliższe informacje można uzyskać pod adresem:
Firma SYSTEM, 85-168 Bydgoszcz, ul. Ujejskiego 48, tel. 715-939.

OD REDAKCJI

Informujemy, że nasze pismo można w dalszym ciągu **zaprenumerować** - co daje pewność systematycznego otrzymywania (drogą pocztową). Nasz miesięcznik kosztuje w prenumeracie 10.000 zł. Prenumeratę można zawrzeć na okres nie krótszy niż dwa miesiące, w dowolnym okresie, maksymalnie do końca roku kalendarzowego. Wykupujący prenumeratę nie ponoszą kosztów przesyłki pocztowej.

Wadliwa dystrybucja „64 plus 4 & Amiga" przez przedsiębiorstwo RUCH jest przyczyną kłopotów, jakie mają czytelnicy chcący nabyć nasze pismo. Zdarza się, że otrzymujemy jako zwroty NIETKNIĘTE paczki zbiorcze. Tymczasem czytelnicy sygnalizują, że do wielu kiosków nie dociera ono wcale. Dlatego:

zapraszamy wszystkich chętnych
do prowadzenia kolportażu
„64 plus 4 & Amiga"
(kluby, studia i sklepy komputerowe, księgarnie,
osoby indywidualne itd.) do współpracy!

Oferujemy korzystne warunki!

Zainteresowanych prosimy o kontakt z działem dystrybucji pod adresem: Przedsiębiorstwo ABUK, 87-200 Wąbrzeźno, ul. 1 Maja 33.



miesięcznik nr 5(19)
maj 1992
cena 1 egz.: 10.000

64 PLUS 4

WYDAWCA: ABUK Spółka z o.o.
REDAGUJĄ: Waldemar Szczygieł (redaktor naczelny) z zespołem.
ADRES REDAKCJI: Redakcja „64 plus 4", 85-166 Bydgoszcz 43, skrytka pocztowa 64.
OKŁADKA: Piotr Bartz.
SKŁAD: ABUK
DRUK: W.Z.G. Wąbrzeźno, okładka: Z.P. POLRASTER, Bydgoszcz.

W numerze :



NOWINKI

W dniach 23, 24 maja br. w Tychach odbył się **V Ogólnopolski Zlot Użytkowników COMMODORE.**

Organizatorem Zlotu była firma handlowo - usługowa IBEX z Tych, powołana do życia przez najaktywniejszych członków byłego Towarzystwa Użytkowników Komputerów: B.K. Grochalskich, K. Kozioła, P. Mięsowicza i G. Stoleckiego.

Na terenie Zespołu Szkół Górniczych PAWK S.A. przy Al. Bielskiej 100 (dzięki uprzejmości dyrekcji szkoły, oraz wydatnej pomocy jej uczniów) spotkało się ponad tysiąc użytkowników komputerów firmy Commodore.

Najliczniej stawili się użytkownicy Amigi. Przyjechała również spora grupa właścicieli C-64. Obecni byli także posiadacze C-16 i plus4 (zaiste, jedyny to przypadek w Polsce przyjęcia tych - niesłusznie odsuwanych w cień - użytkowników na równi z amigowcami).

Impreza została przygotowana bardzo starannie dzięki czemu uniknięto tłoku, awarii sieci zasilającej i tym podobnych zdarzeń, które często mają miejsce na tego typu spotkaniach.

Na miejscu czynny był dobrze zaopatrzony bufet (nie było potrzeby „wyskakiwania gdzieś na miasto"), firma WENCEK ELEKTRONIK wszystkim pechowcom „od ręki" naprawiała sprzęt, na miejscu można było również zarezerwować tani nocleg (dowóz do hotelu - na koszt organizatorów).

Nie zapomniano o odpowiednich tablicach informacyjnych, okolicznościowych plakietkach, długopisach, zapalniczkach...

Podobnie jak w latach ubiegłych tak i tym razem wśród uczestników spotkania rozlosowano nagrody rzeczowe ufundowane przez sponsorów imprezy.

Firmy STD i KD Computer z Katowic ufundowały stację dysków 3.5" do Amigi. Tyski oddział firmy POLFROST ufundował kartę trzydniowego pobytu w dowolnie wybranym hotelu na terenie Europy, firma GigaSof - magnetofon 1530 do C-64 oraz pudełka na dyskietki, firma WIENCEK ELEKTRONIK - zelektronizowany super joystick, firma ACTION z Katowic - zestawy zachodnich czasopism (z dyskami) o amidze, firma TURBO z Mikołowa - dyskietki 3.5", tyski Urząd Miejski - zestawy kaset audio i video, redakcja 64 plus 4 - dwa roczniki naszego pisma i dwa egzemplarze programu D-Mon. Do tego grona dołączyli nawet właściciele bufetu fundując wartościowe bony żywnościowe (bony były okolicznościowe; zapewniano, że nie jest to powrót reglamentacji).

W czasie spotkania siemianowicka grupa fanów amigi przeprowadziła **I Ogólnopolski Turniej KICK OFF**. Była gorąca atmosfera walki, były nagrody (ufundowane przez Polfrost, IBEX, redakcję Amigowca i 64 plus 4), było zabawnie.

Zlot odwiedziła ekipa TVP oraz przedstawiciele prasy.

Dobra organizacja, duża ilość uczestników, różnego rodzaju atrakcje to ważne atuty tej - największej w kraju - imprezy. Do zobaczenia na następnym spotkaniu!

W.S.



UWAGA! Firma **IBEX** z Tych posiada prawa dystrybucji naszych wydawnictw. W razie kłopotów z ich nabyciem dzwońcie: tel. 124-22-76, prosić Piotra.

CHANNEL VIDEODAT - z Astry 1A

Niezwykle interesującą sprawą dla polskich użytkowników komputerów jest połączenie się z siecią komputerową na Zachodzie. Ze względu na fatalny stan naszych linii telefonicznych nie bardzo jest możliwe wykorzystywanie klasycznych modemów. Pojawiła się jednak nowa szansa dzięki wykorzystaniu najbardziej popularnego w Polsce satelity ASTRA 1A. Jeden z programów niemieckich (PRO 7) postanowił zamiast telegazety przekazywać dane komputerowe.

Aby coś „ściągnąć" z satelity trzeba posiadać:

1. zestaw satelitarny ustawiony na Astrę lub KOPERNIKA I (tam gdzie jest nadawany PRO 7) - potrzebny jest tylko sygnał video,
2. komputer IBM, Amiga, Atari ST,
3. specjalny modem satelitarny VD-2000,
4. dyskietkę ze specjalnym programem dla odpowiedniego komputera,
5. dwa kable łączące odpowiednio tuner z modemem i modem z komputerem.

Każdy z użytkowników posiada swój numer i hasło. Każdy z modemów również ma swój numer - wszystkie numery są zastrzeżone i muszą być zarejestrowane w Niemczech. Większość programów jest bezpłatna (jest też oprogramowanie płatne np. jednorazowo 120DM za pół roku lub 180DM za rok).

Dane są przekazywane o konkretnej godzinie dla odpowiedniego komputera (w przyszłości także dla konkretnego użytkownika na numer i hasło).

Cały system jest w ciągłej rozbudowie. Jego wadą jest język niemiecki (a nie angielski) - ma on być w ciągu najbliższego czasu zmieniony na angielski.

Jest to pierwsze powszechne wykorzystanie połączenia komputera z normalną anteną satelitarną.

Bliższe informacje na temat modemu satelitarnego można uzyskać w firmie **HOLLEX, 39-200 Dębica, ul. Świerczewskiego 4/12, tel/fax: ++48 (146) 41-95.**

Sztuczki i kruczki dla C16

C-16

Generator znaków w C16 zawiera 128 znaków (litery, symbole graficzne) oraz 128 tych samych znaków w inwersji. Cały generator zajmuje 1024 bajty (np. od adresu 12288, 13312, 14336, 15360).

W celu zmiany tych adresów wpiszmy:

```
1 POKE 65298, PEEK (65298) AND NOT 4
2 POKE 65299, adres początku gen. znaków/256
```

Poniższy listing umożliwi nam kopiowanie znaków graficznych z ROM'u do RAM'u (pod adres startowy 15360).

```
10    for t=0 to 56
20    read a
30    poke 15000+t,a
40    next
80    sys 1500
62900 data 169,0,141,248,7,133,208,133
62910 data 210,169,208,133,209,169,60,133
62920 data 211,141,19,255,160,0,177,208
62930 data 145,210,200,208,249,230,209,230
62940 data 211,165,211,201,64,208,237,169
62950 data 192,141,18,255,169,59,133
62960 data 52,133,56
62970 data 169,246,133,51,133,55,96
```

Zmiana kolorów na klawiszach

Adresy „klawiszy kolorowych" (CTRL lub klawisz Commodore razem z klawiszami numerycznymi) znajdują się pod adresami od 275 do 290.

Można je oczywiście dowolnie zmieniać np.:

```
poke 275,69
```

Zamiast koloru czarnego pierwszym klawiszem będziemy włączać teraz kolor zielony.

Funkcja samopowtarzania klawiszy

poke 1344,0	:samopowtarzanie wyłączone, funkcję tę mają tylko klawisze CRSR, spacja, INS/DELETE
poke 1344,64	:wyłączenie tej funkcji dla wszystkich klawiszy
poke 1344,128	:samopowtarzanie włączone dla wszystich klawiszy

Bufor klawiatury

Adresy 1319 do 1328 zawierają kody klawiatury, których nie można bezpośrednio odczytać. Komórka 239 zawiera licznik bufora klawiatury, który informuje o ilości kodów złożonych do bufora klawiatury. Oto przykład:

```
poke 1319,13: poke 1320,13: poke 239,2
print "dload": chr$(34)
print "run"
```

Odczyt klawiatury

Pod adresem 198 zostanie zapamiętany kod wciśniętego klawisza. Jeśli żaden nie został wciśnięty to peek(198)=64. Za pomocą tej instrukcji możemy również odczytać wartość klawiszy funkcyjnych. Instrukcją peek (1347) odczytujemy stan klawiszy „shift", „commodore", „ctrl".

Użyteczne komórki pamięci

Wait 1,198 - czeka na wciśnięcie klawisza w magnetofonie.

Pamięć klawiszy funkcyjnych znajduje się w adresach od $0567 do $05e6.

Adresy skoków funkcji USR znajdują się w #1281/1282 czyli $0501/0502.

Basic-start	43/44 (starszy/młodszy bajt)
początek zmiennych	45/46 (j.w.)
początek pola	47/48 (j.w.)
koniec pamięci łańcuchów	51/52 (j.w.)
początek łańcuchów	53/54 (j.w.)
koniec pamięci Basic'a	55/56 (j.w.)

Układ dźwiękowy w C16 można programować w następujący sposób: rejestr dźwiękowy znajduje się pod adresem 65297. Każdemu bitowi odpowiadają określone funkcje sterujące:

bit 0 do 2	=# 0 do # 7, siła głosu = vol 0 do vol 6
bit 3	=#8, pełna siła głosu (jeśli bit 3 jest ustawiony to bity 0-2 są bez znaczenia)
bit 4	=#16, generator 1
bit 5	=#32, generator 2 (1=wł., 0=wył)
bit 6	=#64, generator 2 +szum (=1)
bit 7	=#128, 1 = brak dźwięku

Dźwięk jest kodowany w 10 bitach; możliwa wartość: #0 do #1023.

Częstotliwość	młodszy bajt	starszy bajt (0,1)
gen. 1	65294	65298
gen. 2	65295	65296

Rejestr 65286 - przez skasowanie bitów 0/1 można przesunąć ekran o trzy linie w górę

ustawiony 2 bit	- ekran w dół o cztery linie
skasowany 3 bit	- ekran w dół o cztery linie
skasowany 4 bit	- ściemnienie ekranu
ustawiony 5 bit	- przesunięcie pamięci Char
ustawiony 6 bit	- niewidoczny kursor

Rejestr 65287 -ustawiony bit 0/1przesunięcie ekranu do 3 linii na prawo

ustawiony 4 bit	- włączony tryb wielokolorowy
ustawiony 5 bit	- zatrzymanie układu video



	ustawiony bit 6	- programowalne zakłócenia ekranu
	ustawiony bit 7	- 38 znaków w linii
poke 65290,162		blokuje klawiaturę
poke 65290,163		odblokowuje klawiaturę
Rejestr 65301		kolor tła, bit 0-3 (Low-Nibble) kolor
Rejestr 65305		kolor ramki, bit 4-6 (High-Nibble) intensywność.

Ochrona listowania

A oto niewielki, lecz sprawnie działający program zabezpieczający przed wylistowaniem. Wystartujcie ten program i wpiszcie LIST. Prawda, że działa skutecznie? Procedura ta może znajdować się także w waszych programach, lecz bądźcie ostrożni! Raz uruchomiony program zabezpieczy go przed wylistowaniem na zawsze!

```
10  rem ochrona przed listowaniem
50  for i=1015 to 1055
60  read a$: poke i,dec(a$): ps=ps+dec(a$)
70  next
80  if ps=4182 then poke 774,13: poke 775,4: end
90  print "blad danych"
100 data 0d,91,12,53,4f,52,52,59
110 data 2c,20,4e,4f,20,4c,49,53
120 data 54,20,21,21,21,0d,ea,a9
130 data 00,a2,00,bd,f7,03,20,d2
140 data ff,e8,e0,16,d0,f5,4c,da
150 data 8c
```

PEEK & POKE

poke peek(43)+peek(44)*256-1,1 - przy RUN i NEW komunikuje SYNTAX ERROR. Zachowajcie ostrożność, gdyż rozkaz ten działa pomimo sygnalizacji błędu;
poke peek(43)+peek(44)*256-1,0 - przełączenie do normalnego stanu;
print peek(172) - podaje aktualny, logiczny numer pliku;
print peek(147) - podaje aktualny numer sprzętu;
print peek(200)+peek(201)*256 - podaje początek adresów aktualnej linii obrazowej w Video-Ram'ie. Odejmując od tej wartości 1024, otrzymamy odpowiedni adres kolor -Ram'u.
poke 774,0:poke 775,128 - komputer wykonuje zamiast LIST - RESET.
poke 814,0: poke 815,128 - komputer wykonuje zamiast LOAD - RESET.
poke 816,0: poke 817,128 - to samo przy instrukcji SAVE.
poke 814,164: poke 815,241
poke 816,74: poke 817,240 - zamienione zostaną instrukcje SAVE i LOAD.
wait 1347,1 - czeka aż zostanie wciśnięty klawisz SHIFT.
wait 1347,2 - czeka na klawisz CBM.
wait 1347,4 - czeka na klawisz CTRL.
wait 1347,7 - czeka na jeden z powyższych klawiszy.
poke 1351,128 - blokuje przełączanie przez SHIFT/CBM.
poke 1351,0 - zdejmuje blokadę przez poke 1351,128
poke 2039,1 - blokuje przerwę LIST przez CTRL-S.
poke 2039,0 - odblokowuje powyższą blokadę.
poke 65298, peek(65298)AND 251 - daje znać układowi Video, że powinien skorzystać z pamięci RAM.
poke 65298, peek(65298)OR 4 - daje znać układowi Video, że powinien odczytać dane z ROM'u.
poke 65299, peek(65299)AND 3 OR wartość/255: poke 740,wartość /256 - zmienia adresy startowe znaków graficznych. Zmienna „WARTOŚĆ" musi być wielokrotnością liczby 1024. Jeśli chcecie przesunąć znak graficzny, to musicie poinformować układ Video, że powinien odczytać dane z RAM'u (patrz komórka pamięci 65298).
print peek(65299) AND 252*256 - podaje adres startowy generatora znaków graficznych.
poke 65299,peek(65299)OR 4:poke 740,212 - przełącza duże/małe litery.
poke 65299,peek(65299) and 251:poke 740,208 - przełącza tryb graficzny.
sys 65511 - zamyka wszystkie otwarte kanały i zbiory.
poke 0,0 - załącza silnik magnetofonu
poke 0,15 - i go wyłącza
print peek(1) - odczyt magnetofonu:
200=wyłączony
216=zakłócenia
192=wciśnięty klawisz PLAY/REW/FWD
208=wciśnięty klawisz PLAY i REC
sys 32768 - RESET
poke 240,255 - czeka na naciśnięcie klawisza.
poke 1351,128 - blokada przełączania duże/małe litery.
print peek(1347) - odczyt klawiszy specjalnych.
poke 65286,peek(65286) AND 239 - wyłączenie ekranu.
poke 65286,peek(65286) OR 16 - włączenie ekranu.
poke 814,23 - blokada działania LOAD.

R.G.

ASSEMBLER 6510
lekcja 10

C-64

W poprzednich odcinkach zapoznaliśmy się z podstawami assemblera. Poznaliśmy szesnastkowy system liczenia, tryby adresowania oraz praktycznie wszystkie rozkazy procesora 6510 wraz z większością tzw. rozkazów niepublikowanych. Wspomniałem też o podstawowych sposobach użycia tych rozkazów w prostych procedurach (dodawanie, odejmowanie liczb szesnastobitowych).

Na pewno wszyscy zauważyli, że w assemblerze procesora 6510 nie ma rozkazów służących do wykonywania mnożenia oraz dzielenia. Dzisiaj właśnie zajmiemy się tworzeniem procedur służących do wykonywania tych działań. Będzie to mnożenie dwóch liczb ośmiobitowych oraz dzielenie liczby szesnastobitowej przez ośmiobitową.

Zacznijmy od mnożenia. Przy mnożeniu liczb zapisanych w systemie dwójkowym postępuje się identycznie jak w przypadku liczb dziesiętnych. Oto prosty przykład mnożenia dwóch liczb zapisanych w systemie dwójkowym:

```
    1101
  * 1001
  ------
    1101
   0000
  0000
 1101
--------
 1110101
```

Jak widzimy kolejne składniki sumy są odpowiednio przesuniętym mnożnikiem.

Jak jednak napisać procedurę mnożącą liczby ośmiobitowe w assemblerze? Na początek należy zauważyć, iż wynik takiego mnożenia musi być zapisany w dwóch bajtach, gdyż największa liczba, jaką można zapisać na ośmiu bitach to 255, a 255 * 255 = 65025, na zapisanie wyniku wystarcza szesnaście bitów (czyli dwa bajty). Mnożną, mnożnik oraz wynik działania zapiszemy dla wygody na stronie zerowej. Procedura będzie musiała analizować kolejne bity mnożnika oraz do rejestru wyniku wpisywać odpowienio przesuniętą mnożną. Rozwiązanie to jest jednak niewygodne, gdyż należało by w nim za każdym przebiegiem pętli przesuwać mnożną w lewo (konieczny zapis mnożnej na dwóch bajtach) oraz - w razie takiej konieczności - wykonywać dodawanie do wyniku dwóch bajtów będących odpowiednio przesuniętą mnożną. Jak więc widać rozwiązanie to nie jest optymalne i raczej czasochłonne dla procesora, a przecież w naszych obliczeniach z pewnością będziemy musieli wykonywać szereg różnych skomplikowanych działań, w tym także mnożenia. Warto więc zastanowić się nad pewną optymalizacją podanego algorytmu.

Spróbujmy przeanalizować jeszcze raz mnożenie naszych dwóch liczb czterobitowych. Weźmy pierwszy bit mnożnika. Jest on równy jeden, więc dodajemy mnożną do wyniku działania. Kolejne dwie cyfry to zera, więc wprawdzie mnożnej do wyniku nie dodajemy, jednak przesuńmy wynik za każdym razem o jeden bit w prawo. Tak więc przy wykonywaniu analizy ostatniego bitu mnożnika wykonujemy dodawanie mnożnej do wyniku przesuniętego już o trzy bity. Po wykonaniu ostatniego dodawania także należy przesunąć wynik o jeden bit w prawo i mamy elegancki wynik umieszczony w jednym bajcie (mnożenie dwóch liczb czterobitowych). Oczywiście analogicznie postąpimy w przypadku mnożenia liczb ośmiobitowych, ale będzie trzeba wziąć pod uwagę, iż wynik jest liczbą dwubajtową i będziemy musieli pamiętać o odpowiednim jej przesuwaniu.

Przyjmijmy, że wynik znajduje się w komórkach $FB oraz $FC, mnożna w $FD, a mnożnik w $FE. Procedura będzie wyglądać tak:

```
        LDA #$00     ;Inicjacja wyniku.
        STA $FB      ;
        STA $FC      ;
        LDX #$08     ;Początek pętli.
pętla   LSR $FE      ;Pobranie i sprawdzenie
        BCC skok     ;pierwszego bitu mnożnika.
        LDA $FB      ;Jeżeli testowany bit to
        CLC          ;jedynka, realizowane jest
        ADC $FD      ;dodawanie mnożnej do wyniku.
skok    STA $FB      ;
        ROR $FB      ;Przesunięcie wyniku o
        ROR $FC      ;jeden bit w prawo.
        DEX          ;Zakończenie pętli i skok
        BNE pętla    ;na jej początek lub koniec
        RTS          ;działania procedury.
```

Jak widzimy procedura jest dość krótka i zrozumiała. Jej działanie można jeszcze nieco przyspieszyć przez wpisywanie górnej części wyniku do akumulatora. Dzięki takiej zmianie oszczędzimy kilka jakże cennych cykli, ale za to wynik będzie częściowo zapamiętany w akumulatorze, a częściowo na stronie zerowej. Oto zoptymalizowana wersja procedury mnożącej:

```
        LDA #$00     ;Inicjacja wyniku.
        STA $FB      ;
        LDX #$08     ;Początek pętli.
pętla   LSR $FE      ;Sprawdzenie bitu
        BCC skok     ;mnożnika.
        CLC          ;Dodawania mnożnej
        ADC $FD      ;dowyniku.
skok    ROR          ;Przesunięcie wyniku
        ROR $FB      ;o jeden bit w prawo.
        DEX          ;Zakończenie pętli
        BNE pętla    ;i skok na jej początek
        RTS          ;lub koniec procedury.
```

Jak widać niewiele się różni od pierwowzoru. Mnożna i mnożnik są umieszczone w tych samych komórkach, a wynik mnożenia przechowujemy w akumulatorze (starszy bajt) i w komórce $FB (młodszy bajt).

Proponuję spróbować własnych sił pisząc procedurę mnożącą większe liczby, na przykład dwie szesnastobitowe. Procedura będzie wyglądała niemalże identycznie,

C-64

a warto zasygnalizować, iż możliwe jest wtedy wykorzystanie jej do wykonywania dość skomplikowanych obliczeń na liczbach zmiennoprzecinkowych, które z kolei przydają się w szybkich procedurach do tworzenia grafiki wektorowej...

Przejdźmy teraz do drugiego problemu, jakim jest dzielenie.

Dzielić będziemy oczywiście liczbę szesnastobitową przez ośmiobitową, przy czym dzelnik musi być większy od starszego bajtu dzielnej. Wynikiem dzielenia będzie liczba ośmiobitowa oraz reszta z tego dzielenia. Jak widać poczyniliśmy pewne założenia. Warunkiem działania tej procedury jest to, iż uzyskany wynik nie będzie przekraczał granicy jednego bajtu, i stąd pewne ograniczenia co do dzielnej. Pozornie jest to spore ograniczenie, w praktyce jednak innego rodzaju dzielenia stosuje się niezbyt często.

Na początek proponuję zapoznać się z procesem samego dzielenia liczb w systemie dwójkowym.

Podobnie jak w systemie dziesiętnym dzielenie liczb dwójkowych jest bardzo proste, a dzięki konieczności korzystania tylko z dwóch cyfr nawet łatwiejsze. Oto przykład dzielenia takich liczb:

```
1110
--------
10101011:1100
1100
-----
10010
1100
-----
1101
1100
-----
r.11
```

Aby z dzielenia liczby ośmiobitowej przez czterobitową otrzymać wynik czterobitowy dzielnik musi być większy niż cztery górne bity dzielnej. Taki też warunek będziemy sprawdzać na początku naszej procedury dzielącej. Jak zwykle do przechowania wszystkich danych wejściowych i wyjściowych będzie nam służyć strona zerowa. W przypadku dzielenia potrzebować będziemy jednak pięciu komórek: $FB i $FC będą przeznaczone na dzielną, $FD na dzielnik, do $FE trafi uzyskany wynik, a w $02 umieścimy resztę z naszego dzielenia. Oto procedura:

```
        LDA $FB         ;Sprawdzenie
        CMP $FD         ;poczynionych
        BCS koniec      ;założeń.
        LDA #$00        ;Inicjacja komórek
        STA $FE         ;zawierających
        STA $02         ;wynik działania
        LDX #$08        ;Początek pętli.
pętla   ASL $FE         ;Przesunięcie wyniku.
        ASL $FC         ;Przesunięcie dzielnej
        ROL $FB         ;w lewo i porównanie
        LDA $FB         ;jej starszych bitów
        CMP $FD         ;z dzielnikiem.
        BCC skok        ;Skok warunkowy
        LDA $FB         ;Odjęcie dzielnika
        SEC             ;od starszego bajtu
        SBC $FD         ;dzielnej.
        STA $FB         ;
        INC $FE         ;Jedynka do wyniku.
skok    DEX             ;Koniec pętli skok
        BNE pętla       ;na jej początek lub
        LDA $FB         ;zakończenie programu.
        STA $02         ;Wpisanie reszty do $02.
koniec  RTS             ;Powrót z procedury.
```

Mam nadzieję, że dzięki komentarzom procedura będzie dla wszystkich zrozumiała. Działa ona bardzo podobnie do mnożenia, ale przesunięcie dzielnej odbywa się w lewo. Po dokonaniu przesunięcia wykonujemy porównanie z dzielnikiem, przez które sprawdzamy, czy starszy bajt dzielnej dzieli się przez dany dzielnik. W zależności od wyniku porównania wykonujemy następnie skok albo na koniec pętli, albo do procedury odejmującej dzielnik od aktualnego starszego bajtu dzielnej oraz zapisującej jedynkę do najmłodszego bitu wyniku dzielenia. Czynności te powtarzamy osiem razy, tzn. aż sprawdzimy wszystkie osiem bitów z młodszego bajtu dzielnej. Po skończeniu pętli w starszym bajcie dzielnej otrzymujemy resztę z operacji.

Po dokladniejszym przyjrzeniu się procedurze niektórzy zauważą zapewne niepotrzebne użycie tak dużej ilości komórek ze strony zerowej. Akumulator za to wykorzystywany jest tylko do wykonywania działań na starszym bajcie dzielnej. Pierwszą więc, dość istotną, zmianą w procedurze może być zastąpienie komórki $FB akumulatorem. Z pewnością niewykorzystujemy także w pełni młodszego bajtu dzielnej, który jest stale przesuwany w lewo, podobnie jak wynik. Dlatego też można spokojnie wykorzystać tą komórkę jako bajt, do którego będziemy wpisywać wynik działania. Po wprowadzaniu tych udoskonaleń otrzymamy chyba już optymalną i dość szybką procedurę dzielenia. Do przechowania dzielnej wykorzystamy komórkę $FC oraz akumulator, do komórki $FC wpisywać będziemy również wynik. Pozostaje jeszcze dzielnik, który będzie zapisany w komórce $FD oraz reszta z dzielenia, którą wpiszemy do $FB. Oto krótsza wersja procedury:

```
        CMP $FD         ;Sprawdzenie
        BCS koniec      ;założeń.
        LDX #$08        ;Początek pętli.
pętla   ASL $FC         ;Przesunięcie dzielnej
        ROL             ;w lewo i porównanie
        CMP $FD         ;jej z dzielnikiem.
        BCC skok        ;Skok warunkowy
        SBC $FD         ;dzielnej.
        INC $FC         ;Jedynka do wyniku.
skok    DEX             ;Koniec pętli skok
        BNE pętla       ;na jej początek lub
        STA $02         ;zakończenie programu.
koniec  RTS             ;Powrót z procedury.
```

Nowa procedura jest o wiele krótsza (i szybsza!), lecz mniej przejrzysta. Jednak rozwiązanie to jest dużo bardziej przydatne, gdy będziemy chcieli wykonywać bardziej skomplikowane operacje arytmetyczne, wymagające dużej ilości dzieleń lub mnożeń.

Wydaje mi się, że posiadając przekazane dotąd na łamach naszego pisama wiadomości możemy się pokusić o napisanie pakietu procedur służących do wykonywania czterech podstawowych działań na liczbach szesnasto, lub nawet w razie potrzeby, trzydziestodwubitowych. Możemy także wprowadzić takie działania jak potęga, czy silnia i pokusić się o stworzenie szybkiego programu do wykonywania działań na liczbach z pewnego zakresu. Namawiam więc gorąco do pracy.

Zapraszamy do nadsyłania własnych prac, z których najciekawsze zostaną zamieszczone na łamach naszego pisma. Zapraszam również do nadsyłania propozycji tematów na kolejne lekcje assemblera.

Jarosław "JARRI" Horodecki

SPRITE

C-64

Jednym z najczęściej pojawiających się efektów w demach są różnego rodzaju sposoby poruszania sprite'ów po ekranie. Najczęściej do stworzenia ładnego toru używa się dwóch funkcji sinus. Ale jak to właściwie zrobić? Temu problemowi powięcony jest ten artykuł.

Pracę trzeba oczywiście rozpocząć od procedury inicjującej system przerwań oraz ustalającej wszystkie początkowe dane dla sprite'ów. Procedura ta powinna wyglądać mniej więcej tak:

```
SEI             ;Wyłączenie przerwań.
LDX #$q         ;Ustawienie wektora przerwań
LDY #$irq       ;na procedurę użytkownika.
STX $0314       ;(na adres irq)
STY $0315       ;
LDA #$7F        ;Zabronienie przerwań CIA.
STA $DC0D       ;
LDA #$01        ;Zezwolenie przerwań VIC.
STA $D01A       ;
LDA #$FA        ;Przerwanie zostanie wywołane,
STA $D012       ;gdy wiązka elektronów osiągnie
LDA #$1B        ;dolną ramkę (linia nr. $FA)
STA $D011       ;
LDA #$FF        ;Włączenie wszystkich sprite'ów.
STA $D015       ;
LDA #$00        ;Ustawienie na zero liczników
STA $FB         ;pozycji poziomej i pionowej
STA $FC         ;sprite'ów.
LDX #$3F        ;Przygotowanie danych o kształcie
LDA #$FF        ;sprite'a.
STA $3FC0,X     ;
DEX             ;
BPL wypełń      ;
LDX #$07        ;skok
LDA #$01        ;Ustawienie kolorów sprite'ów
STA $D027,X     ;na kolor biały.
LDA #$FF        ;Ustawienie wektorów kształtów
STA $07F8,X     ;wszystkich sprite'ów na dane
DEX             ;o numerze $FF (od adresu $3FC0).
BPL skok        ;
CLI             ;pętla
JMP petla       ;Zapętlenie.
```

Przy czym pozostałe parametry opisujące sprite'y powinny być wyzerowane (ustawienie standardowe). Nie ma potrzeby wpisywania wartości początkowych współrzędnych poziomych oraz pionowych, ponieważ będą one ustawione przez procedurę poruszającą sprite'y. Wpiszmy jeszcze nową procedurę przerwania (jej adres trzeba podać jako wektor w procedurze inicjującej).

```
irq     INC $D019      ;Przyjęcie przerwania.
        JSR ruch_x     ;Skok do poruszenia w X.
        JSR ruch_y     ;Skok do poruszenia w Y.
        JMP $EA7E      ;Koniec przerwania.
```

Jako podprogram ruchu w osi X (czyli w poziomie) wpiszmy procedurę wpisującą stałe wartości do rejestrów pozycji X.

```
ruch_x  LDA #$18
        STA $D000
        LDA #$30
        STA $D002
        LDA #$48
        STA $D004
        LDA #$60
        STA $D006
        LDA #$78
        STA $D008
        LDA #$90
        STA $D00A
        LDA #$A8
        STA $D00C
        LDA #$C0
        STA $D00E
        RTS
```

Teraz, aby poruszyć osiem sprite'ów w osi Y musimy przygotować specjalną tablicę z danymi otrzymanymi przy pomocy funkcji sinus. Pomoże nam w tym prosty programik w języku basic:

```
10 A=50: B=228: C=256
20 FOR I=0 TO C-1
30 X=SIN((PI/180)*(360*I/C))*((B-A)/2)+((B-A)/2+A)
40 POKE adres1+I,X50 NEXT I
```

Przy czym symbol "pi" należy zastąpić odpowiednim znaczkiem z klawiatury, a zamiast stałej adres1 podstawić wartość dziesiętną adresu, pod którym chcemy umieścić tablicę. Po uruchomieniu program wypełnia obszar pamięci od adresu oznaczonego stałą adres1 odpowiednimi wartościami. Zmienne A oraz B to górny i dolny kres zbioru wartości funkcji, a C długość tablicy. Na początek proponuję skorzystać z podanych wartości wszystkich zmiennych, jednak później możemy dowolnie zmieniać zakresy danych, jak i ich ilość.

Po utworzeniu tablicy funcji sinus musimy oczywiście napisać procedurę, która zajmie się samym poruszaniem sprite'ów (przyjmijmy, że tablica z danymi dla ruchu pionowego znajduje się pod adresem $kkkk).

```
ruch_y  LDX $FB        ;Wczytanie początkowych
                       ;wartości
        LDY #$00       ;do rejestrów indeksowych.
skok1   LDA $kkkk,X    ;Wczytanie danej z tabeli.
        STA $D001,Y    ;Wpisanie danej do rejestru
        TXA            ;pozycji Y sprite'a.
        CLC            ;Zwiększenie znacznika
        ADC #$04       ;aktualnej danej o $04,
                       ;co oznacza odstęp
        TAX            ;$04 pomiędzy kolejnymi
        INY            ;pobieranymi danymi.
        INY            ;Zwiększenie rejstru Y i skok na
        CPY #$10       ;początek pętli, jeżeli warunek
        BNE skok1      ;jest niespełniony.
        LDA $FB        ;Zwiększenie znacznika pozycji
        CLC            ;o jeden (jest to szybkość
        ADC #$01       ;poruszania się sprite'ów.)
        STA $FB        ;
        RTS            ;Wyjście z podprogramu.
```

Po wpisaniu wszystkich tych kawałków i uruchomieniu całości (lepiej najpierw wszystko nagrać) na ekranie ukaże się osiem prostokątów poruszających się w kierunku pionowym według danych dla funkcji sinus... Wygląda to chyba całkiem ładnie.

C-64

Teraz proponuję jeszcze poeksperymentować zmieniając szybkość ruchu oraz odstępy pomiędzy sprite'ami.

Naszym następnym krokiem będzie utworzenie podobnej procedury dla ruchu w poziomie. Tutaj jednakże musimy zadbać o najbardziej znaczący bit pozycji znajdujący się w komórce $D010. Po dokonaniu odpowiednich zmian w programie i po wstawieniu odpowiednich wartości do linii 10 nasz program będzie wyglądał tak:

```
10 A=24: B=320: C=256
20 FOR I=0 TO C-1
30X=SIN((PI/180)*(360*I/C))*((B-A)/2)+((B-A)/2+A)
40 POKE adres2+I,X-INT(X/256)*25645 POKE adres3+I,X/256
50 NEXT I
```

Oczywiście zamiast stałych adres1 oraz adres2 należy wpisać odpowiednie adresy, pod którymi chcemy umieścić tabele dla ruchu poziomego. Teraz należy jeszcze dopisać procedurę, która będzie wpisywała odpowiednie wartości do rejestrów pozycji poziomej sprite'a. Procedura ruchu w pionie będzie bardzo podobna do procedury ruchu poziomego, jednakże należy zwrócić uwagę na rejestr $D010. Za każdym przejściem pętli, zależnie od pobranej z tablicy wartości musimy wykonać instrukcję ORA na rejstrze $D010, a na początku procedury wyzerować go. Oto jak powinna ona wyglądać:

```
ruch_x  LDA #$00      ;Wyzerowania najbardziej
        STA $D010     ;znaczących bitów pozycji X.
        LDX $FC       ;Wczytanie początkowych
                      ;wartości
        LDY #$00      ;do rejestrów indeksowych.
skok1   LDA $nnnn,X   ;Wczytanie danej z tabeli.
        STA $D000,Y   ;Wpisanie danej do rejestru
        LDA $mmmm,X   ;Wczytanie z tabeli 9 bitu.
        BEQ skok2     ;Sprawdzenie, czy 0.
        LDA $llll,Y   ;Pobranie z tabeli odpowiedniej
        ORA $D010     ;maski bitowej dla danego
skok2   STA $D010     ;sprite'a.
        TXA           ;pozycji Y sprite'a.
        CLC           ;Zwiększenie znacznika aktualnej
        ADC #$04      ;danej o $04, co oznacza odstęp
        TAX           ;$04 pomiędzy kolejnymi
        INY           ;pobieranymi danymi.
        INY           ;Zwiększenie rejstru Y i skok na
        CPY #$10      ;początek pętli, jeżeli warunek
        BNE skok1     ;jest niespełniony.
        LDA $FC       ;Zwiększenie znacznika pozycji
        CLC           ;o jeden (jest to szybkość
        ADC #$01      ;poruszania się sprite'ów.)
        STA $FC       ;
        RTS           ;Wyjście z podprogramu.
```

Aby procedura działała prawidłowo należy jeszcze wpisać od adresu $llll następujące liczby:

01 01 02 02 04 04 08 08
10 10 20 20 40 40 80 80

Są to dane dla rejestru $D010 informujące o tym, który bit należy w razie potrzeby ustawić na 1.Oczywiście w pamięci muszą też być zapisane odpowienie wartości w każdej z tabel. I tak od adresu $kkkk - dla współrzędnej Y, od $nnnn - dla współrzędnej X oraz od $mmmm - najstarszy bit dla współrzędnej X.

Pozostaje jeszcze tylko uruchomić gotową całość i poeksperymentować z szybkością ruchu sprite'ów oraz odległościami pomiędzy nimi. Ciekawy efekt da też wprowadzenie różnych prędkości poruszania się dla każdego sprite'a, jednak wymaga to wprowadzenie kilku zmian w samej procedurze.

Jarosław "JARRI" Horodecki

OGŁOSZENIA

KTO PYTA NIE BŁĄDZI

C-64

Codziennie otrzymujemy bardzo wiele listów od naszych czytelników. Nie jesteśmy w stanie na wszystkie indywidualnie odpowiadać. W tym artykule postaram się odpowiedzieć na najczęściej pojawiające się w nich pytania.

1. Czy programy demonstracyjne umieszczane na Public Domain Pack'ach można wylistować nie posługując się żadnym specjalnym programem?

Niestety nie. Aby poznać sposób realizacji efektów, które są prezentowane we wszystkich programach demonstracyjnych musimy posłużyć się programem monitora języka maszynowego. Dostępnych jest bardzo wiele tego rodzaju programów, według mnie jedne z lepszych to EX-MON, C-MON oraz Mega Monitor. Wadą ich jest to, iż każdorazowo przed użyciem należy je wczytywać do pamięci komputera, co powoduje zniszczenie pewnego jej obszaru, w którym może znajdować się kod. Dlatego też moim zdaniem najlepszym rozwiązaniem jest nabycie jednego z popularnych modułów do C64 jak Final II lub III czy Action, z których ten ostatni ze względu na wiele przydatnych udogodnień jest chyba najlepszy.

2. Czy można użyć programu Voicetracker v4.0 lub Music Searcher do tworzenia podkładu muzycznego we własnych programach demonstracyjnych?

Oczywiście programy te można w wyżej wymienionym celu wykorzystać. Voicetracker to program służący do tworzenia własnej muzyki (lub przerabiania gotowej), natomiast Music Searcher służy tylko i wyłącznie do "wyciągania" utworów muzycznych z różnych programów demonstracyjnych oraz niektórych gier.

3. Czy istnieje program, którym można kopiować samouruchamiające się programy zapisane na dysku?

Oczywiście, że tak. Istnieje bardzo wiele takich programów, z których stanowczo bezkonkurencyjny jest Fast Hacke'em. Przy korzystaniu z jednej stacji wykonuje on kopię jednej strony dyskietki w czasie około 2 minut. Jeżeli mamy dwie stacje to kopiowanie bez weryfikacji trwa tylko 35 sekund, a z weryfikacją 55 sekund. W pakiecie tym znajdują się również specjalne kopiery służące do wykonywania kopii zabezpieczonych przed dalszym kopiowaniem.

Ostatnio w Polsce stało się również popularne podłączanie stacji dysków do C64 przez interface równoległy. Przy użyciu specjalnych programów kopiujących możliwe jest wtedy skopiowanie strony dysku w ciągu zaledwie 15 sekund!

4. Czy istnieje jakiś program służący do przegrywania programów z magnetofonu na dysk?

Z tego, co wiem to opcję taką posiada program Spectacular Copy. Wadą jego jest brak zainstalowanego systemu turbo dyskowego, co znacznie spowalnia proces kopiowania. Wygodne jest natomiast to, że program automatycznie określa przewidywaną długość w blokach kopiowanego programu i w razie braku miejsca na dysku sygnalizuje to podaniem odpowiedniego komunikatu.

5. Kupiłem na giełdzie program Atari Emulator, ale nie wiem jak go uruchomić. Po jego wgraniu ukazuje się napis "Atari - emulator (c)1984 by Pro-data" oraz pytanie "using Rs 232 (y/n)". Po dowolnej odpowiedzi na ekranie ukazują się kolorowe pasy, a program każe kontynuować wgrywanie.

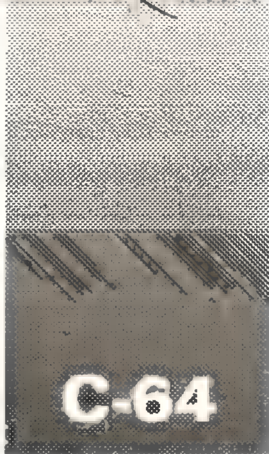
Już dość dawno spotkałem się z tym programem, jednak ja również nie mogłem go uruchomić i z tego co wiem w Polsce nie istnieje działająca jego wersja. Być może program ten jest po prostu czyimś żartem lub też wymaga posiadania dodatkowych plików, które nie dotarły do Polski. Przy okazji, jeżeli ktoś posiada działającą wersję tego programu to proszę o kontakt z redakcją.

6. Przy pomocy programu Music Searcher "wyciąłem" muzyczkę z dema. Co zrobić, aby użyć ją we własnym programie demonstracyjnym?

Rozumiem, że demo zostało napisane w języku maszynowym i pytanie dotyczy odegrania muzyczki z poziomu tego właśnie języka. Zanim odpowiem na pytanie może mała dawka wiadomości.

Istnieje na C64 bardzo wiele różnych tak zwanych player'ów czyli programów odtwarzających napisaną muzykę według danych zapisanych w pamięci. Większość z tych procedur, aby ułatwić ich używanie, ma na początku umieszczone zwykle dwa lub trzy skoki typu JMP $nnnn, z których zawsze jeden jest odpowiedzialny za inicjację procedury, a drugi za odegranie jednego dźwięku.

Aby muzyczkę można było odegrać należy w procedurze przygotowującej system do potrzeb naszej demonstracji dopisać rozkaz skoku (JSR $nnnn) pod adres pierwszego JMP procedury odtwarzającej, który zwykle odpowiada procedurze inicjującej nasz utwór. Często należy skok ten poprzedzić zapisaniem wartości $00 do jednego z rejestrów



procesora. Czasem może się rownież zdarzyć, iż za inicjację muzyczki odpowiedzialny będzie drugi lub też trzeci skok. Drugą rzeczą, jaką musimy wykonać jest wstawinie skoku (też JSR) do drugiego JMP procedury odtwarzającej w dowolne miejsce naszej procedury przerwania. Skok ten odpowiada zwykle za odegranie pojedynczego dźwięku. Oczywiście w przypadku tego skoku również zdarzają się wyjątki, tak więc gdy po wstawieniu odpowiednich skoków muzyczka nadal nie chce grać musimy trochę poeksperymentować z wstawianiem różnych kombinacji adresów skoków.

7. Co zrobić, aby muzykę ułożoną przy pomocy programu Voicetracker można było odegrać z poziomu interpretera języka basic?

Możliwość taka oczywiście istnieje i jest opisana w instrukcji do programu. Mianowicie procedura odtwarzająca została wyposażona w krótki programik, który może odegrać cały utwór. Działanie jej można przerwać w każdej chwili przez naciśnięcie klawisza SPACE, a aby ją uruchomić należy wykonać instrukcję SYS 4102.

8. Jak wydrukować tekst z poziomu języka maszynowego?

Bazując na tym jednym pytaniu można spokojnie napisać kilkustronnicowy artykuł. Jednak ze względu na ograniczoną objętość naszego pisma omówię jedynie najprostszy sposób rozwiązania tego problemu.

Otóż wpiszmy najpierw do pamięci prosty programik w języku maszynowym:

```
      LDX #$00     ;
pętla LDA $nnnn,X  ;Pobranie znaku z tablicy.
      JSR $FFD2    ;Wydrukowanie znaku na ekranie.
      INX          ;Zwiększenie licznika o jeden
      CPX #$mm     ;oraz po sprawdzeniu warunku
      BNE $pętla   ;skok na początek pętli lub...
      RTS          ;KONIEC
```

Ta prosta procedurka umożliwi nam wydrukowanie tekstu o maksymalnej długości 256 bajtów zapisanych w pamięci od adresu $nnnn przy pomocy kodów ASCII. Oczywiście zamiast $nnnn musimy wpisać odpowiedni adres, a zamiast $mm ilość znaków, jaką chcemy wydrukować na ekranie.

9. W jaki sposób można wykorzystać pamięć RAM w Commodore 64 znajdującą się pod pamięciami ROM?

Jak wiadomo procesor 6510 ma szynę adresową szesnastobitową, co daje mu możliwość adresowania maksymalnie 64kB pamięci. C64 obok 64kB pamięci typu RAM posiada jeszcze 20kB pamięci ROM, w których zapisany jest system operacyjny komputera oraz intepreter języka basic i dwa generatory znaków. Kroje liter widoczne są tylko przez VIC'a, więc nie przeszkadzają w używaniu pamięci. Natomiast interpreter basic'a oraz system umieszczone są "nad" pamięcią RAM i przesłaniają do niej dostęp. Jedyne, co możemy z poziomu basic'a to wpisywać w ten obszar dane przy pomocy instrukcji POKE, ale bez możliwości ich późniejszego wykorzystania. Przy odpowiednim przekonfigurowaniu rejestrów VIC'a możemy również umieścić w tych obszarach pamięć ekranu lub też wzory sprite'ów, gdyż VIC nie widzi w tych miejscach ROM'u ale zwykły RAM. Pełne wykorzystanie przez procesor tych obszarów pamięci możliwe jest tylko z poziomu języka maszynowego, z tym, że bardzo rzadko korzysta się z możliwości odłączenia ROM'u z systemem operacyjnym, gdyż procedury w nim umieszczone są często bardzo przydatne w tworzeniu własnych programów.

Sterowaniem pamięciami zajmuje się jeden rejestr znajdujący się pod adresem 1 o następującej architekturze:

bit 0 - jeżeli ustawiony na 1 to włączony jest ROM basic'a, jeżeli na 0 to wyłączony;

bit 1 - analogicznie jak bit 0, ale dotyczy ROM'u z systemem;

bit 2 - jeżeli 1 to od adresu $D000 do $E000 znajduje się obszar rejestrów wejścia/wyjścia, jeżeli 0 to w obszarze tym znajduje się generator znaków, który może zostać odczytany przez procesor;

bity 3-5 - odpowiedzialne za obsługę magnetofonu;

bit 6-7 - nie podłączone.

Warto jeszcze zwrócić uwagę, że jedyną modyfikacją jaką możemy dokonać z poziomu interpretera języka basic jest zmiana bitu 2, po której możemy przepisać generator znaków do pamięci RAM i dokonać w nim dowolnych zmian. Zmiany bitów 0 oraz 1 spowodują natychmiastowe zawieszenie systemu. Można jednak przepisać zawartość pamięci ROM, zawierającej interpreter basic'a, do pamięci RAM znajdującej się pod nim i odłączyć pamięć ROM. Umożliwia to wprowadzenie dowolnych zmian w interpreterze, począwszy od zmiany raportów oraz słów kluczowych, a skończywszy na poprawieniu błędów basic'a.

10. Czy po wykonaniu freeze w module FINAL III i przejściu do monitora istnieje możliwość powrotu do wykonywanego programu? Jakie są różnice pomiędzy modułami ACTION oraz FINAL III?

Niestety nie. Jedynym modułem, jaki jest wyposażony w taką opcję jest ACTION. Tak więc jedyne co pozostaje użytkownikowi FINAL'a III to próba znalezienia adresu startowego programu.

Co do różnic pomiędzy modułami FINAL III oraz ACTION to są one dość duże. Począwszy od ceny, a skończywszy na możliwościach. Po pierwsze moduł ACTION jest przeznaczony dla użytkowników już nieco oswojonych z komputerem. Posiada wiele narzędzi ułatwiających programowanie w języku maszynowym przy dość słabym rozszerzeniu basic'a. FINAL III natomiast ma bardzo dobrze rozszerzony interpreter basic'a i jest bardzo wygodny i łatwy w obsłudze. Wszystkie jego opcje dostępne są ze specjalnego menu składającego się z okienek, którymi można dowolnie poruszać po ekranie. ACTION natomiast wszystkie opcje każe wstukiwać z klawiatury. W porównaniu z FINAL'em III, ACTION posiada też nieco bardziej rozbudowany monitor języka maszynowego oraz więcej przydatnych programistom możliwości w menu freeze. Tak więc, podsumowując to krótkie porównanie, należałoby polecić ACTION'a osobom, które pragną zająć się komputerem poważnie, natomiast FINAL'a III tym, którzy stawiają raczej na BASIC oraz zabawę.

Jarosław "Jarri" Horodecki

OPIS ZESTAWU PUBLIC DOMAIN PACK

nr 15 (dysk - kwiecień '92)

C-64

Oto mamy przed sobą kolejny Public Domain Pack. Znajdziemy na nim kilka ciekawych programów użytkowych, demonstracyjnych oraz ostatnie numery polskich magazynów dyskowych dla C64.

CIABAH 4&5 to podwójny numer szczecińskiego magazynu wydawanego przez grupę Crazy Boys. Znaleźć w nim możemy kilka ciekawych artykułów, np.: reportaż z wyjazdu członków grupy na warszawskie party, ostatnie wiadomości z polskiej i zagranicznej sceny, a także opisy kilku programów użytkowych oraz nowych gier.

Drugim magazynem zamieszczonym na naszym PDP jest **Darkside** grupy Parados wydawany przez jej Tarnowską sekcję. Tym razem autorzy podzielili go na dwie części, co umożliwiło zamieszczenie większej ilości artykułów. W numerze znajdziemy kolejną relację z warszawskiej imprezy. Oprócz tego możemy zapoznać się dokładniej z programem Turbo Assembler v5.1, oraz z najnowszymi wieściami o polskiej i zagranicznej scenie.

Na dysku znajduje się również demo grupy Beyond Force pod tytułem **"Part Trap"**. Zawiera kilka nowych efektów np.: ciekawie zrealizowany efekt "gwiazdek", rozciąganie napisów w poziomie, duży scroll sinusowy, bardzo efektowny scroll z kropek poruszający się po różnych torach, kula zrobiona z 370 kropek obracających sie wokół trzech osi. Demo to zostało wydane na świątecznym party zorganizowanym przez Szwedzką grupę Light.

Drugie demo to kolejna produkcja grupy Skylight. Jest to zestaw obrazków nowego grafika grupy o pseudonimie Liar. Biorąc pod uwagę niezłą jakość obrazków jest to z pewnością obecnie jeden z najlepszych grafików w Polsce.

Pozostały jeszcze programy użytkowe. Pierwszy z nich to **Convert Studio**, niemieckiej grupy Padua. Program umożliwia zamianę prawie dowolnego obrazka na format czterokolorowy (w przypadku trybu multicolor) lub też zwykły dwukolorowy w wysokiej rozdzielczości (w przypadku obrazka w trybie hi-res). Możliwa jest także konwersja obrazków na generator znaków wraz z odpowiednią mapą, przy czym możliwe jest zarezerwowanie dowolnych znaków. Ciekawym rozwiązaniem jest możliwość użycia myszy od Amigi zamiast zwykłego joysticka. Do programu dołączono także dość dokładną instrukcję w języku angielskim.

Kolejny program to nowy produkt szczecińskiej grupy Skylight: edytor znaków w wysokiej rozdzielczości o wielkości trzy na trzy piksele. W programie możliwe jest korzystanie z dowolnej mapy znakowej definiowanej przez użytkownika. Dzięki takiemu rozwiązaniu możliwe jest otrzymanie nawet 64 znaków. Program posiada wiele ciekawych opcji podzielonych na kilka menù (umieszczonych na głównym ekranie roboczym). Wadą jest z pewnością nieco zagmatwany ekran główny.



Możliwe jest wykonanie inwersji oraz odbicia lustrzanego zarówno pojedynczych znaków jak i całego zestawu. Oczywiście można także nagrać przygotowane znaki na dysk w formie generatora wraz z mapą fontów. Autor programu wykonał również procedurę umożliwiającą stworzenie prostego scrolla przy użyciu narysowanych fontów. Przy wykonywaniu operacji wejścia/wyjścia można korzystać zarówno z taśmy jak i z dysku.

Możliwe jest oczywiście ustalanie własnej mapy znaków, ale program rozpoznaje także standard ustalony w edytorze węgierskiej grupy Faces. Często przydaje się też możliwość zgrania aktualnie rysowanego zestawu znaków do bufora.

Sprite Designer to już drugi program użytkowy grupy Skylight. Umożliwia on rysowanie małych obrazków na ośmiu spriteach ustawionych w dwóch rzędach po cztery spritey w każdym. Wszystkie opcje programu zawarte są w małym menu umieszczonym w lewym dolnym rogu ekranu. Program jest wykonany dość starannie i praca z nim jest dość łatwa. Jedyną, moim zdaniem wadą, jest przydzielenie na edycję obrazka zbyt małej części ekranu pozostawiając sporą jego część wolną. Być może autor przewiduje przyszłą zmianę programu na zwykły edytor do obrazków w trzech kolorach...

Ostani już program grupy Skylight to **Multicolour Converter v1.3** umożliwiający zmianę kolorów dowolnego obrazka wykonanego w technice wielokolorowej.

Następny program pod tytułem **Logo Flipper** to produkt grupy Chrome. Umożliwia on wykonywanie dwóch operacji na trójkolorowych obrazkach zapisanych w formacie Logo Editora Centauri. Możliwe jest przekształcenie obrazka, aby był on wyświetlany "do góry nogami" lub też wykonanie odbicia lustrzanego.

Ostatni z programów użytkowych to **Charset Maker v1.0** napisany przez Walta ze znanej grupy Bonzai. Służy on do konwersji zestawu znaków narysowanego przy pomocy edytora służącego do rysowania obrazków w trzech kolorach na format znaków używany w scrollach. Program ten przydatny jest zwłaszcza wtedy, gdy chcemy użyć fontów o wielkości przekraczającej standardowe wymiary (tzn. większe niż 16 na 16 pikseli). Aby użyć Charset Makera należy najpierw przy pomocy dowolnego programu graficznego narysować potrzebny nam zestaw znaków. Następnie przy pomocy dowolnego konwertera należy zamienić nasz zestaw na generator znaków wraz z mapą ekranu i w takiej formie wgrać go do Charset Makera. Przy pomocy opcji "Edit Charset Datas" należy ustawić rozmiar naszego zestawu znaków oraz ilość znaków jaką chcemy w nim umieścić. Teraz, korzystając z opcji "Enter Charset" wycinamy kolejne literki i umieszczamy je pod odpowiednimi klawiszami. Gdy już wytniemy wszystkie znaki możemy obejrzeć rezultat naszej pracy oglądając specjalnie do tego celu przygotowany scroll. Po wybraniu opcji "Save Finished Charset" program podaje kilka niezbędnych do późniejszego użycia tychże znaków infromacji.

Jarosław "JARRI" Horodecki

DLA WSZYSTKICH

„MODEMOWANIE" cz. 2

W poprzednim numerze poświęciliśmy jeden z artykułów sprawom telekomunikacji oraz sposobów korzystania z tajemniczych pudełek zwanych modemami. Dzisiaj spróbujemy nieco przybliżyć ten temat naszym czytelnikom podając kilka informacji przydantych każdemu chcącemu zająć się „modemowaiem".

Jeszcze jakiś rok lub dwa lata temu kupowanie i późniejsze używanie modemu było rzeczą bardzo prostą i nie wymagającą głębszego zastanowienia. W grę wchodziły w zasadzie tylko modemy 1200 oraz 2400 bodów. Ich ceny były do siebie bardzo zbliżone, a większość była zgodna z uznanym przez wytwórców standardem Hayes. Niestety, ostatnio z powodu coraz większego zapotrzebowania i coraz większej ilości wytwórców sprzętu sytuacja zmieniła się. Wprowadzane są różne nowe standardy, ciągle też zachodzą zmiany w szybkości przesyłania danych, które na zwykłych liniach telefonicznych, przy sprzyjających warunkach dochodzą nawet do 38400 bodów! Oprócz znacznego przyspieszenia transmisji nowe modemy są także wyposażone w szereg udogodnień, jak sprzętowa korekcja błędów czy też kompresja danych. Wszystkie te dodatki mają swoje nazwy. I tak często słyszymy określenia: V.32, V .32bis, V.42, V.42bis, MNP-4, MNP-5 i wiele innych niezrozumiałych skrótów.

Wszytkie te skróty pochodzą od nazw trzech podstawowych cech modemów: sposobu modulacji, kompresji danych oraz kontroli błędów. Na początek przyjrzyjmy się dokładniej sposobom modulacji, jakie stosują najbardziej popularne modemy telefoniczne.

Jak wiadomo z artykułu w poprzednim numerze słowo modem oznacza MOdulator/DEModulator. Modem zamienia cyfrowe sygnały otrzymywane z komputera na sygnały analogowe wysyłane do linii telefonicznej, lub też sygnały analogowe na ich odpowiedniki cyfrowe, gdy dane otrzymuje. To właśnie sposoby zamieniania sygnału cyfrowego na analogowy i odwrotnie są nazywane protokołami modulacji. Różne protokoły determinują sposób kodowania danych oraz prędkość ich przesyłania. Zwykle każdy modem ma conajmniej kilka standardowo wbudowanych protokołów modulacji. Modemami typu High-Speed (o dużej szybkości transmisji) nazywamy modemy, które pozwalają na osiągnięcie realnej prędkości pracy od 9600 bodów w górę. Modemami tego typu nie są modemy 2400 mogące przy pomocy kompresji sprzętowej teoretycznie osiągnąć prędkość 9600 bodów.

Pozostaje jeszcze pytanie co oznacza spotykany często skrót CCITT? Otóż jest to skrót od francuskiej nazwy agencji Organizacji Narodów Zjednoczonych do spraw telekomunikacji. Organiazacja ta zajmuje się między innymi ustalaniem norm dla komunikacji modemowej.

Jak już wspomniałem, modemy dzielimy na szybkie (od 9600 bodów) oraz zwykłe (2400 bodów lub wolniejsze). Na początek zajmijmy się normami ustalonymi dla tych właśnie modemów. Każdy z nich zasadniczo powinien oferować następujące sposoby transmisji danych przez łącza telefoniczne:

Bell 103 - umożliwia przesyłanie z prędkością 300 bodów, jest to standard amerykański.

Bell 212A - przesyłanie danych z prędkością 1200 bodów, także standard amerykański.

CCITT V.22 - prędkość przesyłania 1200 bodów, standard przyjęty poza terytorium USA.

CCITT V.22bis - prędkość 2400 bodów, standard międzynarodowy.

Niektóre modemy zapewniają także następujące protokoły transmisji:

CCITT v.21 - prędkość 300 bodów, norma dla innych krajów niż USA.

CCITT V.23 - prędkość 1200/75 bodów lub 75/1200 bodów, tzn. dane są wysyłane do użytkownika z prędkością 1200 bodów, a otrzymywane z prędkością 75 bodów, jest to standard europejski i jest używany przez różnego rodzaju publiczne biuletyny informacyjne.

W przeszłości modemy 2400 bodów nie oferowały żadnych udogodnień, obecnie jednak sytuacja się zmieniła i modemy te posiadają szereg funkcji dodatkowych jak kompresja danych czy - bardzo przydatna w polskich warunkach - sprzętowa korekcja błędów.

Po krótkim opisie modemów 2400 przyszedł czas na szybkie modemy. Oto krótka charakterystyka stosowanych w nich standardów transmisji danych:

V.32 - jest to protokół ustalony dla modemów o prędkościach przesyłania informacji 9600 oraz 4800 bodów.

CCITT V.32 został przyjęty już w 1984 roku, jednak dopóki ceny ich były zbyt wysokie (do 2000 dolarów) modemy tego rodzaju nie zostały przyjęte przez rynek. Teraz jednak można kupić już dobry modem oferujący pracę z prędkością 9600 bodów oraz szereg innych udogodnień za cenę nie przekraczającą 500 dolarów. Taki spadek cen tych modemów wynika z tego, iż każdy producent został przez konkurencję zmuszony do wprowadzenia na rynek własnego produktu spełniającego normę V.32. Nawet firmy produkujące modemy wyposażone we własne sposoby transmisji zwykle włączją do gamy swoich wyrobów modemy typu "dual-standard" (posiadające zarówno własne protokoły jak i V.32).

V.32bis - standard ten został przyjęty na początku 1991 roku i jest ustalony dla modemów o prędkości przesyłania danych 14400 bodów. Modemy te mogą także pracować z prędkością 12000, 9600, 72000 oraz 4800 bodów w razie pogarszania się stanu linii telefonicznej. Należy także pamiętać, iż standard V.32 jest zgodny ze standardem V.32bis (oczywiście modemy wyposażone w V.32 nie mogą racować z prędkością powyżej 9600 bodów).

Oprócz standardów określających sposób przesyłania danych istnieją także normy ekreślające sposoby sprzętowej korekcji bledów w przesyłanych danych, a także ich sprzętowego pakowania.

V.42 i MNP-4 - są to standardy określające sposób kontroli błędów. MNP jest skrótem od nazwy Microcom Networking Protocol i został opracowany przez firme Microcom. Istnieją protokoły korekcji błędów o oznaczeniach od MNP-2 do MNP-4, natomiast MNP-5 jest protokołem kompresji danych.

Norma V.42 została ustalona przez CCITT. Mimo, że norma ta jest bardziej rozbudowana niż MNP-4 modemy wyposażone w korekcję błędów V.42 mogą współpracować z modemami posiadającymi tylko korekcję na poziomie MNP-4. Sprzętowa korekcja błędów służy w głównej mierze do zaniedbywania przekłamań pojawiających się w wyniku różnego rodzaju szumów powstających na łączach telefonicznych. Dzięki pracy modemów wyposażonych w korekcję błędów na ekranie nie musimy oglądać tzw. krzaków,

PUBLIC DOMAIN PACK

PUBLIC DOMAIN PACK C-64

Styczeń '91 (nr 1)

- Mega demo grupy „VISION"-MIST2, Preview do gier: UN SQUADRON, PUZZLENOID, TURRICAN.

Luty '91 (nr 2)

- Tune of month, Logo Writer V 2.0, Fast Cruelcrunch, WRATH+ (DEMO)[02], Dreptacz - BASIC, SWISS CHEESE/CFA, Disk Fast Loader.

Marzec '91 (nr 3)

- Font Grub 1.0, Projektant Duszków, Strzałka 64+, Piratek - gra, V4.0 - Symphonies, Cruiser, The First, Commercial Break, Relakator 64, Korektor 64, Flash, HOT SHOT nr 9 (zach. mag.), BAD NEWS nr 2 - j.w., demo - rekord - 290 sprite'ów!, demo: NEW INTRO, demo: LET'S DYCP, Kontakt Corner - adresy, New Fast - działa z 1541 i 1541 II, CSLINKER V2.0.

Kwiecień '91 (nr 4)

- Digi - Organizer - program do tworzenia muzyki z użyciem digitalizacji dźwięku, „ONE YEAR - RADIUS" - mega demo grupy RADIUS.

Maj '91 (nr 5)

- CRUEL SOLIDERS - demo, DESTINATION - demo, SUCKER DJ! - demo (digi mix), MUSIC SEARCHER - do wycinania ilustracji muzycznych z programów, MEGA DEMO „INFOSYSTEM 91".

Czerwiec '91 (nr 6)

- Fonteditor, Sindata Editor, Color Editor, Disk - Noter, Gwiazdy - demo graficzne, FILGRAEPH 2.2/BML, NOTE TO FLI V 2.2, AFLI - EDITOR V 1.2, RESET - MON,8,1, TURBO - ASS 5, HIGHLIFE #5, AXEL NEWS #1, DISK NOTKA/PADUA, PSC - MAG #9'06/91, CONSPIRE? OREGON - demo, CONTACT DEMO/ORE, SHOWPIX.

Lipiec '91 (nr 7)

- Mega demo „MY, OH MY!" grupy LIGHT, Game Music Composer - edytor muzyczny grupy GRAFFITY z Węgier.

Sierpień '91 (nr 8)

- MegaDemo „Unnamed" grupy CAMELOT, Sound Killer - edytor muzyczny grupy TOPAZ, AFLI - edytor graficzny techniki A-FLI, Disk-Dos obsługa komend stacji dysków, Noter v2.2 grupy TOPAZ, IFFL - Squeezer kompresor dyskowy, Dismaster+ - edytor do dyskietek, Super Copy - DOS szybki program kopiujący do zbiorów, Mega Demo fińskiej grupy TOPAZ - „Graveyard Blues".

Wrzesień '91 (nr 9)

- Mega Demo grupy FLASH, Hot Shot - magazyn dyskowy, Code Sucker monitor - pr. użytkowy grupy PADUA, Mountain Ride - gra w BASIC.

Październik '91 (nr 10)

- MEGA DEMO „AIRDANCE 4" grupy T.A.T.

Listopad '91 (nr 11)

- NEW LAW & ORDER, FLT/LEGOLAND, FLT/LEGONOTE, TERMINAT .2%/FLT, SM. CRIMINAL #8, SMALL BUT FINE, HOLLY SMOKE/M12, UNITE!/SYLVIO.

Grudzień '91 (nr12)

- Armageddon 3, NOTE TO DEMO, OUTRUN 2 MUS $ SFX, AFTER-BURNER/MON, TRIVIA-GAME MUSIC, FORM.I. SIMULATOR, 2400AD END-TUNE, NIGHTHUNTER MUSIC, TOMCAT MUS./MON, ZAMZARA TUNE/MON, NOTE TO DISK, HIGHLIFE #9.

Styczeń '92 (nr 13)

- Char Zoomer v3.1, Colour Bar Editor v3.0, Hires+A-FLI Editor, FLI Designer v1.1, Accesinus, Music Routine Cruncher v1.5, Gandalf Protector v3.0, Gandalf Coder v1.0, Disk-tape copy v.20, Gnd - packer v1.0.

Luty '92 (nr 14)

- LYNX XVI+, Sideborder Logo Editor V1.0, Intelpaint, Flieditor V3.2, 4*4 Charmaker, F(R)ONT EDITOR 3, THE GRAFIX PACK II, DEMA.

Marzec '92 (nr 15)

- Contact Dealer v3.0, Beeftrucker v1, Cross linker v3, PowerCruncer v7.1, Sample mon v.2.0, Handy Term v8.4, Highlife, Darkside, Humor Basic, dema.

Kwiecień '92 (nr 16)

- CIABACH 4&5, DARKSIDE, Convert Studio, edytor znaków grupy Skylight, Sprite Designer, Multicolour Conventer v1.3, Logo Flipper, Charset Maker v1.0.

PUBLIC DOMAIN PACK AMIGA

Styczeń '91 (nr 1)

- Programy kompresorów danych, Grafiki Borysa Vallejo, prezentacja najlepszych muzyczek, INTUITRACKER.

Luty '91 (nr 2)

- Request player; Multi ripper, 3-rd day; Phantasmagoria - demo, Master Seka; Virus Ekspert v1.6, AMOS-programy; Moduły: Killing game show, Upon Me, Let's swing it.

Marzec '91 (nr 3)

- PROTRACKER V1.0 (pakiet programowy), Najlepsze muzyczki: NOW WAIT? - DR.AWESOME, AMOS - procedury, DEMO grupy REBELES „TOTAL TRIPLE TROUBLE".

Kwiecień '91 (nr 4)

- RUBBER VECTORS - demo, KEF-TALES - demo, DISK MASTER V3.0, Moduły muzyczne: TECHNOSTYLE 2, GALAXY 2, GRAFIKA - prezentujemy rysunki - RICK PARKS.

Maj '91 (nr 5)

- VIRUS X 5.0, VIRUS TERMINATOR, PARADOX - demo, STORMCHILD - demo, Moduły muzyczne: MIAMI VOICE, ANTI ATARI SONG.

Czerwiec '91 (nr 6)

- POWER BOOT - własne menu dysku, DISK CODING SYSTEM - program do zabezpieczania dysków, Konwerter IFF - ANSI, AUER NATION - demo, Moduły muzyczne, DOCS - opis gry ELWIRA, LAMER DEFENCE - do wykrywania i niszczenia wirusów, REWENG GO OF THE LAMER - grafika w trybie D-HAM.

Lipiec '91 (nr 7)

- Sanity - demo, Amiga - Tanx (1Mb) - gra, Little Beau (1MB) - gra, There is A Light/Tonid - modules.

Sierpień '91 (nr 8)

- Real 3D - demo programu do raytraceing'u, moduł muzyczny XTC STEREO.

Wrzesień '91 (nr 9)

- Moduły muzyczne dla programu TFMX: R - TYPE; The House of Techno;VIRUS EXPERT v181 + 143; Boot Block'i, Bootx v 3.80, Imploder v 4.0.

Październik '91 (nr 10)

- ANARCHY - „THE INSPIRATION IS NONE"; „DUAL CREW - „NEW DIMENSION"; SANITY - „ELYSIUM".

Listopad '91 (nr 11)

- COMPUTER HEAD - animacja; CONFUSED - moduły pod medplayer i wiele sampli; ROCKED -; SAVE GAME „MONKEY ISLAND".

Grudzień '91 (nr 12)

- GEM X; BOOTX V4.13; FINAL KIT - monitor; MEGA-MON; VARIA.

Styczeń '92 (nr13)

- Super Duper 2.01; Sanity Copy; Noise Packer 3.00; Ham Sharp; Mostra; dema graf. i muz.

Luty '92 (nr 14)

- Nuke Saddam 1.4; THIEF RIPPER 2.0; DISK MASTER 3.05; ZIG ZAG #3 (grafika); dema graf. i muz.

Marzec '92 (nr 15)

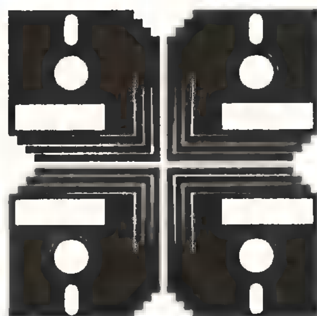
- Gry: KIM, Power Wars; PrtDrvGen; File-Master v1.1; FixDisk!; Facc; Intro Well!; digitalizacje (IFF).

Kwiecień '92 (nr 16)

- Disk Cruncher; IBM-ST-AMIGA; Ham-Lab v0.91; BAD v4.13; BOOTX v4.47,; DragonTiles - gra; grafiki IFF.

PUBLIC DOMAIN PACK





PUBLIC DOMAIN PACK C-64
TAPE NR 1

- TURBO
- SINUSDATA - EDITOR
- FAST CRUNCHER V3
- ANAL S.C. IBEYOND
- VECTOR - VICTORY
- PUZZLENOID+4
- TUNE OF MONTH #1
- NIM
- STRZAŁKA 64+
- LOGO - WRITER V.2.0
- CAN'T TOUCH IKU!
- NTRO PRV
- BONZIEED!!
- ZAX PACKIS
- READ THIS FIRST
- COMMERCIAL BREAK
- 290 SPRITES!
- NOTE - ABOUT
- BAD NEWS NR2
- TO BAD NEWS...
- CONTACT CORNERI
- PROJEKT DUSZKÓW
- SYMPHONY NR14
- SYMPHONY NR15
- SYMPHONY NR16
- SYMPHONY NR17
- SYMPHONY NR18
- SYMPHONY NR19
- CRUISER/GIANTS
- NOTE>ANO<PADUA
- LET'S DYSPI
- FINALTAPE
- MUSIC - SEARCHER

PUBLIC DOMAIN PACK C-64
TAPE NR 2

- TURBO
- PUBL. DOMAIN. INFO
- FONTGRUB 1.0
- DREPTACZ BASIC
- LOAD DIS FIRSY
- MACROASSEMBLER
- TURBOASSEMBLER
- RELOCATOR
- LOGOPAINTER 3!
- REASSEBLER
- SPRITE - EDITOR
- FAST - CRUEL U.2.5
- HIGHLIFE NR5
- AXEL NEWS NR1
- GWIAZDY
- FLIGRAPH 2.2/BML
- NOTE TO FLI V.2.2
- DISKNOTKA/PADUA
- MEGA PACKER/T
- MIST II/ VISION
- TTECHSCR &DYSP
- PLASMA - WORLD
- VECTORBOBS...
- VECTOR - PLOTS
- FLI - UPSCROLL
- BORDER - HIRES
- ROCK AROUND
- FACEWRITER
- CHAR EDIT 2+2
- DISKNOTER
- DESTINATION'91
- CONTACTDEMO/ORE
- FONTEDITOR
- THE END

PUBLIC DOMAIN PACK C-64
TAPE NR 3

- TURBO
- PUBLIC DOMAIN NOTE
- GRAVEYARD NOTES!
- NOTE FROM BEATI!
- ANONYM SPEAKINGI
- SNDK. V3.7/TOPAZ
- AFLI - EDITOR
- NOTER V2.2/TOPAZ
- DLW V1.5/TOPAZ
- CODE - S.MON/PADUA
- OPINION - POLL/PDA
- MOUNTAIN RAID
- PART 1
- PART 2
- PART 3
- PART 4
- PART 5
- FAIRLIGHT 1
- FAIRLIGHT 2
- FAIRLIGHT 3
- FAIRLIGHT 4
- FAIRLIGHT 5
- THE END

PUBLIC DOMAIN PACK C-64
TAPE NR 4

- TURBO
- OUT RUN 2 MUS & SFX
- AFTER BURNER/MON
- FORM.1.SIMULATOR
- 2400 AD.END - TUNE
- NIGHT HUNTER DIGI
- ELEMINATOR MUSIC
- TOMCAT MUSIX/MON
- ZAMZARA TUNE
- DYNAMIX TUNE
- HIGHLIFE NR9
- SNAKES C3
- SNARK C3
- SNERD C3
- WAREHOUSE C3
- STARTREK C3
- TOWER
- SNOOPY
- NEW LAW & ORDER
- FLT/LEGONOTE...
- TERMINAT.2%/FLT
- UNITE!/SYLVIO
- BALL - SCOPE/451
- TRIVIA - GAME MUS.
- RESET - MONITOR
- HOLY SMOKE

Zestawy „64 plus 4 PUBLIC DOMAIN PACK" można zamawiać wpłacając na konto: Bank PKO SA Oddział w Bydgoszczy konto nr: 5.09011-400522.7-2511-30-111.0 następujące kwoty: 20.000zł za pojedyńczy zestaw dyskowy dla C-64, 30.000 zł za zestaw programów PD na kasecie, 25.000zł za zestaw dla Amigi.

Blankiety wpłat powinny być CZYTELNIE wypełnione i zawierać: **imię i nazwisko, dokładny adres zamawiającego, skrót „PDP-64D"** - jeśli zamawiamy zestaw dla C-64 na dyskietce lub **„PDP-64T"** - dla zestawu taśmowego, zestaw dla Amigi prosimy zaznaczać skrótem **„PDP-A"** - dane te prosimy umieszczać na **wszystkich** odcinkach dowodu wpłaty.

W prenumeracie zestawy kosztują: **PDP-64 - 18.000zł** (12 numerów 216 tys. zł), **PDP-A - 22.000 zł** (12 numerów 264 tys. zł). Prenumeratę można zawrzeć w dowolnym terminie na okres od 3 do 12 miesięcy (do końca roku kalendarzowego). Powyższe warunki odnoszą się również do naszych zestawów wydanych w 1991r .

Zestawy taśmowe PDP-64 w 1992r. będą ukazywały się w miarę napływu nowych, ciekawych programów - o czym będziemy informować na łamach naszego pisma.

Zamów nie zwlekaj!

VOICETRACKER V4.0

C-64

Rewelacyjny program muzyczny!



Tylko 50.000 zł kosztuje fantastyczny edytor muzyczny wykorzystujący ogromne możliwości dźwiękowe komputera Commodore - 64. Oferowany zestaw zawiera dyskietkę lub taśmę magnetofonową z programem VOICETRACKER V4.0, instrukcję obsługi, oraz - dodatkowo - przykładowe demonstracje muzyczne.
UWAGA! Wersja magnetofonowa tylko 40.000 zł!

Przedsiębiorstwo ABUK posiada wyłączność na dystrybucję tego programu. Wszelkie kopiowanie programu i powielanie instrukcji jest zabronione. Nabywcy otrzymują rejestrowane kopie programu wraz z prawem nabywania nowych wersji po znacznie obniżonych cenach oraz wymiany dyskietki w razie uszkodzenia. Studiom komputerowym proponujemy zakup hurtowy (przy zakupie powyżej 10 kompletów udzielamy 20% rabatu).
Chcąc stać się posiadaczem programu VOICETRACKER V4.0 wystarczy dokonać wpłaty 50.000 zł (wersja dyskowa) lub 40.000 zł (taśma) na konto: Bank PKO SA Bydgoszcz, konto nr: 5.09011-400522.7-2511-30-111.0. Na blankiecie prosimy czytelnie podać swoje imię, nazwisko i adres wraz z dopiskiem „VV4.0" uzupełnionym literką „T" - taśma lub „D" - dyskietka.

W związku z pojawiającymi się kłopotami w dystrybucji oferowanych przez nas dyskietek i taśm (wynikającymi z nieczytelnego bądź niekompletnego wypełnienia blankietów wpłat) przedstawiamy obok specjalny druk. Blankiet ten może służyć jako zamówienie i dowód wpłaty dla wszystkich oferowanych przez nas usług: sprzdaż dyskietek i taśm PDP, Voicetracker'a, zamówienie ogłoszeń itd. Informujemy, że urzędy pocztowe zobowiązane są do przesłania dwóch odcinków (dla banku i dla wpłacającego) do naszego banku.

REDAKCJA

Odcinek dla wpłacającego

Zł ....
słownie ....
....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-2511-30-111.0.

Opłata
zł..........

Odcinek dla posiadacza rachunku

Zł ....
słownie ....
....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-2511-30-111.0.

Opłata
zł..........

Odcinek dla Poczty

Zł ....
słownie ....
....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-2511-30-111.0

Opłata
zł..........



TREŚĆ ZAMÓWIENIA: ....

TREŚĆ ZAMÓWIENIA: ....

Prosimy o CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA: ....

Prosimy o CZYTELNE wypełnienie.

Zapraszamy wszystkich do udziału w stałym konkursie pod hasłem:

Najlepszy program miesiąca

W konkursie udział mogą brać wszyscy, którzy nadeślą własne, nigdzie nie publikowane prace. Tematyka programów dowolna. Konkurs rozgrywany jest osobno dla komputerów C-64 i Amiga. Teksty programów należy nadsyłać na adres redakcji na dyskietce lub w postaci czytelnego rękopisu (dyskietki będą przez redakcję zwracane). Objętość programu wraz z opisem i komentarzem nie powinna przekraczać 4 stron maszynopisu. Raz w miesiącu Sąd Konkursowy wybierze najlepsze programy przyznając ich autorom dwie główne nagrody po

500.000 zł każda.

Decyzje Sądu Konkursowego są nieodwołalne. Oprócz zdobycia głównej nagrody autorzy mają szansę na publikację swych prac na łamach naszego pisma. Pracę prosimy podpisywać imieniem i nazwiskiem oraz dokładnym adresem autora.

Redakcja

WSZYSTKICH ZAINTERESOWANYCH NABYCIEM ZALEGŁYCH NUMERÓW

„64 plus 4 & AMIGA"

INFORMUJEMY, ŻE POSIADAMY JESZCZE OGRANICZONĄ ILOŚĆ NUMERÓW OD LISTOPADA 1990R. DO GRUDNIA 1991R.

ZAMÓWIENIA PROSIMY KIEROWAĆ NA ADRES :
Przedsiębiorstwo ABUK sp. z o.o.,
87-200 Wąbrzeźno,
ul. 1 Maja 33.
(Pod tym adresem mieści się dział kolportażu - tam też prosimy przesyłać wszelką korespondencję dotycząca kolportażu czasopisma, dyskietek, taśm itd. Adres redakcji się nie zmienił - patrz stopka.)

czyli znaków ukazujących się w wyniku zakłóceń. Sprawdzanie poprawności odbywa się na zasadzie porównania pewnych wskaźników dla danych otrzymanych oraz wysłanych. W przypadku, gdy dane te się nie zgadzają przesłanie ich jest powtarzane aż do uzyskania pełnej poprawności.

Wyższość sprzętowej korekcji błędów nad korekcją stosowaną przez protokoły transmisji plików polega na tym, iż jest ona włączona stale podczas całego połączenie, a nie tylko podczas przesyłania danego pliku.

Inną zaletą protokołów V.42 oraz MNP-4 jest to, iż umożliwiają one wyrzucanie bitów stopu oraz początku każdego bajtu. To znaczy zamiast wysyłać dla każdego bajtu 10 bitów wysyłane jest tylko 8, co zmniejsza całkowitą ilość przesyłanych danych o jakieś 20%. Jednakże z powodu zawsze istniejących choć w minimalnym stopniu zakłóceń na linii nie jest możliwe osiągnięcie optymalnej prędkości przesyłania. W większości przypdków okazuje się także iż podczas przesyłania plików spakowanych protokół V.42 mimo większej kompresji powoduje nieznaczny spadek prędkości w porównaniu z MNP5. Wynika to z tego, iż operacjia ta zajmuje odpowiednio więcej czasu w modemach wyposażonych w V.42.

Czy w takej sytuacji potrzebne są protokoły korekcji błędów? Niestety, są one konieczne nawet w przypadku stosowania prędkości 2400 bodów. Wynika to ze złego stanu central oraz linii telefonicznych, a wydaje mi się, że nikt nie lubi pojawiających się niezrozumiałych znaczków psujących cały układ ekranu.

Pozostały nam jeszcze standardy sprzętowego pakowania danych. Protokołami takimi są MNP-5 oraz V.42bis. Protokoły te współpracują ze sprzętową korekcją błędów i mimo, że istnieją modemy nie wyposażone w sprzętowe pakowanie, to modemów z samym tylko pakowaniem nie produkuje się. Tak więc aby mógł działać MNP-5 modem musi być wyposarzony w MNP-4 i analogicznie aby działało V.42bis modem musi posiadać protkół V.42. Pierwszy z tych protokołów w najlepszych warunkach zapewnia kompresję 2:1, czyli na przykład modem 9600 wyposażony w ten sposób kompresji może teoretycznie przesyłać dane z prędkościa do 19200 bodów. V.42bis może natomiast wykonywać kompresję w stosunku 4:1. Z tego powodu często widzimy na modemach 9600 informacje o możliwości ich pracy z prędkością 38400. Jest to jednakże prędkość nieosiągalna w przypadku zwykłych, spakowanych już wcześniej plików. Jak już pisałem protokoły te mimo podawanych możliwości pakowania dwu lub nawet czterokrotnego zwykle niewiele dają, a często mogą nawet powodować spowolnienie transmisji. Dzieje się tak dlatego, iż realna jakość kompresji zależy od przesyłanych danych. Jeżeli na przykład chcemy wysłać lub też pobrać nie spakowany plik tekstowy to możemy otrzymać prędkość często zbliżoną do 38400 bodów (dla V.42bis i modemu 9600 bodów). Jednakże jeżeli wykonujemy upload lub też download pliku spakowanego przy pomocy jednego ze znanych archiwizerów jak LhArc czy Arj czy PkZip nie należy oczekiwać dobrej prędkości, gdyż często pakowanie i ponowne rozpakowywanie przez oba współpracujące modemy kolejnych pakietów danych może opóźniać transmisję. Mimo więc możliwości stosowania kompresji sprzętowej znacznie lepszym rozwiązaniem jest stosowanie wcześniej wykonywanej kompresji przy pomocy programów specjalnie do tego celu napisanych. Ze stosowania takiego rozwiązaania wynika wiele zalet.

Po pierwsze skompresowane pliki zajmują więcej miejsca na dysku, po drugie większość programów kompresujących pozwala na łączenie kilku plików lub nawet całych podkatalogów w jeden plik, dzięki czemu aby pobrać na przykład cały program składający się z kilkunastu plików wystarczy wskazać jeden, w którym zawarte są wszystkie potrzebene pliki. W końcu kompresja przy pomocy programów jest czestokroć dużo bardziej wydajna i wyraźnie widoczna w przypadku kompresji długich plików. Z praktyki wynika, że przesyłanie plików wcześniej spakowanych jest około 30% szybsze niż pakowanych przez modem.

Przy protokołach kompresji warto jeszcze wspomnieć o sposobie porozumiewania się modemów.

Przypuśćmy, że używamy sprzętowej kompresji danych na poziomie V.42bis oraz modemu pracującego z prędkością 9600. Tak więc maksymalna możliwa do osiągnięcia prędkość to 38400 bodów i na taką właśnie prędkość musimy ustawić port naszego komputera. Podczas pobierania plików i pracy z samym systemem nie ma żadnych problemów, jednak pojawiają się one przy próbie wykonania uploadu (wysłania pliku).

Gdy komputer wysyła dane do modemu z prędkością 38400 bodów, a modem może je wysłać z maksymalną prędkościa 20000 bodów to co się dzieje z nadmiarem danych? Otóż modem musi w jakiś sposób poinformować komputer o konieczności przerwania transmisji i oczekiwania na sygnał o możliwości dalszego jej rozpoczęcia. Właśnie do tego celu służy kontrola lokalnego przepływu danych (local flow control).

Zwykle modemy wyposażone są w dwa jej rodzaje: sprzętowej kontroli (RTS/CTS) oraz programowej (XON/XOFF). Przy czym iepszą z nich jest metoda RTS/CTS. Metoda ta polego na zmianie napięć na liniach RTS (request to send) oraz CTS (clear to send) złącza RS232 pomiędzy modemem oraz komputerem. CTS używane jest przez modem, aby poinformować komputer o gotowości do przyjęcia danych. Po jego odebraniu komputer rozpoczyna wysyłanie kolejnego pakietu danych do modemu. Gdy modem nie może już przyjmować danych z narzuconą przez komputer prędkością po prostu wyłącza sygnał CTS, co informuje komputer o zapełnieniu bufora modemu. Komputer natychmiast zatrzymuje wysyłanie danych i czeka na otrzymanie kolejnego pozwolenia.

RTS ma dokładnie to samo działanie, z tym, że nakazuje modemowi zatrzymanie przesyłania danych do komputera aż do ponownego włączenia tego sygnału.

Wspomniałem wcześniej także o spobie programowego porozumiewania się pomiędzy modemem i komputerem. Otóż polega on na wysyłania w odpowienim momencie znaków kontrolnych odpowiadających kodom XON oraz XOFF. Są to po prostu kody control+Q oraz control+s. Zastosowanie tego sposobu może spowodować pewne nieporozumienia gdy przesyłany plik zawiera kody tychże znaków. Często mogą one nawet doprowadzić do przerwania transmisji lub, co jest chyba znacznie gorsze, do przesłania uszkodzonego po drodze pliku.

opracował na podstawie dokumentacji z różnych BBS'ów - Jarosław "Jarri" Horodecki

AMIGA

FILE MASTER V1.1

Ostatnio zauważyłem, że część moich kolegów nie umie obsługiwać programu File Master poprawnie i dlatego preferują starsze i gorsze programy. Skłoniło mnie to do napisania tego artykułu.

File Master jest programem pozwalającym na wszelkiego rodzaju operacje na plikach, umożliwia obsługę dysku, formatowanie itp. Program podczas działania zajmuje prawie cały ekran i trzeba powiedzieć, że wykorzystuje go bardzo efektywnie.

Na ekranie znajdują się dwa okna do wyświetlania katalogów, pomiędzy którymi znajduje się kolumna z komendami. Niżej znajdziemy informację o ilości wolnej pamięci chip oraz fast, godzinę, datę, dwa małe gadżety oraz cztery komendy.

Gadżety te służą do:

- " - włączony spowoduje wyświetlanie katalogu podczas czytania z dysku.
- ! - gdy jest włączony to program będzie pytał czy zapisać np. nową wersję programu o tej samej nazwie jeszcze raz, kasując starą. Gdyby wykrzyknik był zgaszony to komputer skasowałby starą wersję zastępując ją nowszą.

Dalej znajdują się komendy: SIZE, S-D, MENU, SLEEP, QUIT.

SIZE - podaje zsumowaną wielkość wybranych plików.

S-D to zwykłe "Copy source to destination" czyli skopiowanie zawartości aktualnego OKNA (okna, a nie dysku!) do drugiego.

MENU - ponieważ istnieje więcej komend, niż może się zmieścić w kolumnie na środku ekranu, więc ta opcja służy do ich przełączania.

SLEEP - zmienia File Mastera z aktywnego na program "drzemiący", pojawi się wtedy małe okienko na desktopie, dzięki któremu możemy ponownie go zaktywizować (trzeba kliknąć prawym klawiszem myszy).

QUIT - to może chyba pozostać bez wyjaśnienia.

No dobrze, ale co z komendami w środku? Oto dokładne wyjaśnienie:

DRIVES - wybór dysku (Volumes), urządzenia (Devices) lub katalogu systemowego (Drawers). To właśnie w Devices pojawi się NW0: i dlatego korzystanie z NW0: przy pomocy File Master'a jest tak wygodne (jeżeli nie wiesz co to takiego NW0: to przeczytaj najpierw opis do PDP-AMIGA a wszystko stanie się jasne). Aby wyświetlić katalog dysku należy kliknąć na jego nazwę prawym klawiszem myszy.

PARENT - właściwie nie wymaga wyjanień.

INVERT - zmienia podświetlenie wszystkich plików w aktualnym oknie.

CLEAR - gasi wszystkie podświetlone pliki w aktualnym oknie.

COPY - kopiowanie pilków i katalogów z aktualnego okna do drugiego. Jeżeli zaznaczysz katalog to zostanie on skopiowany razem ze wszystkimi swoimi podkatalogami i plikami. Jeżeli jakiś program już istnieje to F.M. zapyta czy ma go zastąpić nowym (oczywiście tylko wtedy, gdy zapalony jest gadżet - ! -).

MOVE - działa podobnie do COPY, jedyną (ale ważną!) różnicą jest to, że zaraz po skopiowaniu plik zostanie skasowany (komputer nie będzie o nic pytał!).

DELETE - skasowanie zaznaczonych plików i katalogów w aktualnym oknie. Jeżeli zaznaczysz katalog to zostanie on skasowany ze wszystkimi podkatalogami i plikami.

RENAME - zmiana nazwy pliku lub katalogu. Aby zmienić nazwę dysku należy najpierw wybrać DRIVES a następnie podświetlić nazwę dysku i RENAME aby ją zmienić.

MAKEDIR - założenie nowego katalogu.

SHOWASC - czytanie pliku w formacie ASCII.

SHOWHEX - czytanie dowolnego pliku.

SHOW PIC - oglądanie obrazka zapisanego w formacie IFF. Ta opcja ma niestety dwie wady. Po pierwsze nie pokazuje obrazków w trybie OVERSCAN, po drugie podczas oglądania obrazka w trybie INTERLACE jeżeli ruszy się myszą, z niewyjaśnionych powodów zwiększa się migotanie obrazu.

PLAY MOD - odtwarzanie modułu, z któregoś z trackerów.

DISKINFO - informacja o „płaskokrążku informacjonośnym", który akurat znajduje się w elektronicznym czytniku płaskokrążków informacjonośnych (stacji dysków, ha!, ha!).

PROTECT - umożliwia zmianę wszystkich znaczników podświetlonego pliku (znaczniki te to: H S P A R W E D).

EXECUTE - umożliwia uruchomienie pliku wykonywalnego.

COMMENT - pozwala na dopisanie komentarza do pliku (do 80 znaków).

To wszystkie komendy z menu numer jeden, chciałbym zwrócić uwagę, że większość komend jest taka sama w innych programach typu File Master (np. DiskMaster 3.05, 1.4, 1.6,II; SID; DirectoryOPUS).

Teraz komendy z drugiego menu: dwie pierwsze mają znaczenie identyczne jak poprzednio - DRIVES i PARENT. Nowe to:

FORMAT - pozwala na sformatowanie dysku DF0:, DF1:, DF2:, czy też DF3: z weryfikacją lub bez.

QFORMAT - szybkie formatowanie (tylko zerowa i czterdziesta ścieżka). Uwaga! opcji QFORMAT można używać tylko w przypadku, gdy dysk został już kiedyś sformatowany normalnie! Polecam jednak tę opcję gdyż - jeżeli przypadkiem tak sformatujemy potrzebny nam dysk - to będziemy mogli pliki odzyskać spowrotem za pomocą jednego z wielu służących do tego celu programów. (Być może na następnym Public Domain Packu uda się umieścić jeden z nich).

DECRUNCH - umożliwia rozpakowanie pliku spakowanego PowerPackerem.

MODINFO - informacja o module, jego długości, autorze, ilości sampli itp.

FILEEDIT - uruchamia świetny edytor plików. Ma on całkiem duże możliwości i jest bardzo szybki.

Obsługa edytora:

1. Klawisze kursora - gdy kursor nie jest widoczny umożliwiają przesuw edytowanego pliku w lewo i prawo o jedną linię, w górę i w dół - o jeden ekran. Jeżeli kursor jest widoczny to możemy się nim poruszać po części pliku wyświetlonej na ekranie.

2. Gadżety.

Strzałka do góry - przesunięcie do góry o 22 linie (588 bajty). Analogicznie za strzałką do dołu.

Podwójna strzałka do góry - skok na początek pliku.

Podwójna strzałka do dołu - skok na koniec pliku.

Pos: - skok do podanej lini.

3. FIND - szukanie ciągu znaków.

4. CONT - kontynuacja poszukiwań.

5. UNDO - usunięcie dokonanych zmian.

6. WRITE - zapis zmienionego pliku na dysk.

7. EXIT - wyjście z edytora (plik nie zostanie zmieniony).

COLORS - automatyczna zmiana palety kolorów.

DOCS - bardziej szczegółowy opis programu - po angielsku.

MESSAGE - od autora.

ABOUT - o autorze.

I jeszcze uwaga - jeden z moich znajomych nie umiał założyć ram dysku pod File Masterem (co za lamerstwo!) a więc podaję jak to zrobić: pod każdym oknem do wyświetlania katalogu znajduje się linijka, w której program wyświetla aktualną ścieżkę. Wystarczy po prostu napisać w tym miejscu RAM: i return - to wszystko.

Myślę, że po takim kursie absolutnie nikt nie powinien mieć problemów z używaniem File Mastera. Przypuszczam też, że sporo osób zacznie go używać. I bardzo dobrze bo trzeba przecież sobie ułatwiać życie, a nie na odwrót!

Michał "Amber" Gosztyła

PrtDrvGen

W ostatnim opisie Public Domain Pack'u obiecałem, że w następnym numerze znajdzie się opis do PrtDrvGen'a. Obietnicy dotrzymuję - oto dokładny opis funkcji i możliwości tego użytecznego programu.

I. Programu PrtDrvGen tak naprawdę składa się z dwóch części:

1. PrtDrvGen:
- interfejs użytkownika pozwalający wprowadzać komendy i parametry dla drukarki;
- pozwala edytować już istniejący driver;
- zgrywa wprowadzone komendy i parametry do pliku (nazwa driver'a-.dat), który jest zapisany w formacie ASCII i może być wydrukowany;
- program potrzebuje pliku PDG24.txt dla wszystkich wyświetlanych tekstów (wygeneruje plik indeksowy dla szybszego dostępu, PDG24.idx)

2. PrtDrvGen2: - jest automatycznie uruchamiany z PrtDrvGen'a, ale może także zostać uruchomiony ręcznie;
- odczytuje komendy i parametry z pliku nazwa driver'a.dat;
- wczytuje szkielet driver'a z pliku PrtDrv.lod;
- zapisuje gotowy, wygenerowany driver do pliku nazwa driver'a bez rozszerzenia

II. Jak uruchomić program?

- PrtDrvGen musi zostać uruchomiony z CLI i będzie używał plików z aktualnego katalogu.

- Na dysku PDP-Marzec (15) znajduje się plik skryptowy (PrtDrvGen-WB), używa on programu IconX, który pozwala uruchomić PrtDrvGen'a z Workbench'a.

- Nie trzeba wpisywać żadnych komend podczas uruchamiania PrtDrvGen'a. Wystarczy tylko wpisać nazwę (dla CLI), bądź też kliknąć na ikonę PrtDrvGen-WB, która uruchomi plik skryptowy poprzez IconX'a.

- Po uruchomieniu PrtDrvGen'a, zostaje uruchomiony automatycznie PrtDrvGen2. Możesz uruchomić PrtDrvGen'a2 oddzielnie, ale nie ma żadnej potrzeby aby tak robić na Amidze z 512 Kb ramu.

- Zawsze pamiętaj o wciśnięciu klawisza RETURN po zmianie lub wprowadzeniu lini komendy dla drukarki. Jeżeli tak nie zrobisz, komenda nie zostanie zapamiętana.

- Nigdy nie umieszczaj żadnych spacji w nazwie driver'a.

- Jeżeli wybierzesz opcję "Save parameters only" gdy wychodzisz z programu wszystkie aktualne linie parametrów zostaną zapisane do pliku z rozszerzeniem ".dat". Użyj tej opcji jeżeli chcesz wydrukować plik".dat".

III. Konieczne parametry

- Plik Sample.dat zawiera parametry dla drukarek NEC CP6 (P6,P7,CP7,P5XL) kompatybilne z 24-igłowymi, kolorowymi drukarkami Epsona.

- Parametry dotyczące wysyłania tekstu przez driver to 1-3 i 6-226 (130-226 definiują rozszerzone znaki ASCII, które nie są konieczne. Jeżeli nie są to zaznacz zero w 130). Dla samego tekstu musisz przynajmniej ustawić następujące parametry:
3, 6-13, 16-17, 20-21, 24-25, 28-29, 45-46, 130. Następnie ustaw: 44, 58, 60, 68, 98-99, 108.

- Większość pozostałych parametrów dotyczy grafiki. Parametry 230-257 są powtórzone jako 258-285, 286-313 i 314-341 tylko dlatego, że możliwe są cztery różne tryby graficzne. Poniżej zostaną przedstawione tylko 230-257, ale stosuje się to do wszystkich czterech grup.

- Minimum parametrów dla grafiki to 227, 230, 232, 234-239, 247, 253, ale sprawdź dokładnie w instrukcji do swojej drukarki czy nie są potrzebne jeszcze inne.

- Użyj parametru 242 (odpowiednio

AMIGA

270, 298 i 326) aby zmienić rozmiary pionowe obrazka.

Parametry komend dla drukarki oraz parametry ^p i ^dDriver pozwalają na dostęp do 8 parametrów/wartości.

Parametry ^p0 - ^p3 (^d0 - ^d3) pozwalają na dostęp do komend drukarki Amigi.

Parametry ^p4 - ^p7 (^d4 - ^d7) są wartościami wewnętrznymi. Parametry te pozwalają na transfer danych z

a) standardowych komend drukarki Amigi

b) wartości wewnętrznych driver'a do komend drukarki, które chcesz wysłać. Prosty przykład to komenda SLRM (ustawienie lewego i prawego marginesu): jeżeli chcesz ustawić marginesy na 5 i 75 dajesz komendę ESC[5;75s. Zakładamy, że twoja drukarka rozumie tę komendę, więc napisałbyś tak w polu komend (dla SLRM) w PrtDrvGen'ie:

^[[^d0;^d1s

Aktualny ciąg wysłany do drukarki byłby:

ESC[005;075s"^[

to po prostu inny sposób zapisywania znaku ESC, który znajduje się w każdej komendzie dla drukarki Amigi.

Pownieneś pamiętać, że ^d0 jest pierwszym parametrem komendy, ^d1 następnym itd. Parametry przyjmują wartości od 0 do 255 (bajt), ale zapisane wewnętrznie w driverze są jako wartości szesnastobitowe ze znakiem (od -32768 do 32767), które możesz używać w połączeniu z parametrami ^a oraz ^s. Parametry zaliczone do komend drukarki są zaznaczone w PrtDrvGen'ie.

Jeżeli żadne parametry nie są wymagane znaczy to, że wartość parametrów "^p" i "^d" jest niezdefiniowana dla parametrów od 0 do 3. Parametry od 4 do 7 będą posiadały ostatnio zapamiętaną w nich wartość. Użyj parametru ^f0^nnn aby ustalić format ciągu generowanego przez komendę ^d, jeżeli początkowa wartość trzech znaków nie odpowiada twoim potrzebom.

Użyj ^f0^242 (242 ósemkowo to 162 dziesiętnie) aby ustalić podstawowy format dwóch znaków heksadecymalnych.

Dane graficzne w buforze mogą być zapisane bajt po bajcie jako:

a) Kolejne kolumny, od lewej do prawej:

Bajt pierwszy: [(0,0), (0,1), (0,2) .... (0,7)]

Bajt drugi: [(1,0), (1,1)...(1,7)]

b) Kolejne kolumny, od prawej do lewej:

Bajt pierwszy: [(0,7), (0,6), (0,5) .... (0,0)]

Bajt drugi: [(1,7), (1,6)...(1,0)]

c) Kolejne rzędy, od lewej do prawej:

Bajt pierwszy: [(0,0), (1,0), (2,0) .... (7,0)]

Bajt drugi: [(8,0), (9,0)...]

d) Kolejne rzędy, od prawej do lewej:

Bajt pierwszy: [(7,0), (6,0), (5,0) .... (0,0)]

Bajt drugi: [(15,0), (14,0)...]

Parametry dostępne przez komendy ^p, ^d, i ^w podczas druku grafiki to:

```
   Parametr "0-1"        Parametr "2-3"
[xxxxxxxx xxxxxxxx] [xxxxxxxx xxxxxxxx]
    ^p0      ^p1        ^p2      ^p3
     binary bytes       ^d0      ^d1
    ^d2      ^d3      [illegible] digit bytes
        ^w0                ^w2
     [illegible] digit words
```

Parametr "0-1" = ^p0 * 256 + ^p1 = ^w0

Używanie drivera. Program obsługujący drukarkę w AmigaDOS i specyficzny kod drukarki (drivery dla drukarek) nie będzie załadowany do pamięci dopóki nie zostanie wykonana pierwsza operacja wyjściowa do drukarki. Zapewne już przekonałeś się, że po załadowaniu drivera czasem nic się nie dzieje. Często jest to spowodowane brakiem pamięci dla buforów tworzonych przez driver, lub dlatego, że marginesy ustawoine w Preferences'ie są zbyt „ciasne" dla drukowanego obrazka. Długość kodu drivera jest (ze względu na dodane możliwości) większa niż innych driverów, ale jeżeli zabraknie mu miejsca zredukuje swoje zapotrzebowanie na pamięć i automatycznie przełączy się w tryb co prawda wolniejszy ale i oszczędniejszy. Jeżeli mimo wszystko nie starczy pamięci wydrukuje na drukarce napis "No mem" (podając ilość jaką potrzebuje, zapisaną heksadecymalnie). Możesz tego uniknąć ustawiając ciaśniej marginesy w Preferences'ie. To spowoduje wydrukowanie mniejszego obrazka, ale zredukuje zapotrzebowanie na pamięć. Możesz także ustawić drukowanie z mniejszą gęstością (jeżeli twoja drukarka posiada kilka trybów gęstości druku (dots per inch-dpi)).

Program PrtDrvGen jest programem shareware co oznacza, że aby otrzymać pełną jego wersję należy za niego zapłacić. Adres autora: J. Thomsen Kirsebarhaven 14 DK-4000 Roskilde DENMARK

opracował dla was
Michał "Amber" Gosztyla

AMIGA

Instrukcja IF w Amiga-Basic służy do sterowania programem w zależności od wyników porównań. W kilku wypadkach można z niej zrezygnować.

Interpreter Basic'a umożliwia porównuje wartości: jeśli wynik porównania jest logiczną prawdą, to będzie przyjęta wartość -1:

4 > 3

jest prawdziwe gdyż 4 jest większe od 3.

Wypróbujcie to. Napiszcie (trybie bezpośrednim):

PRINT 4 > 3

Otrzymamy wartść -1.

Wprowadźcie:

PRINT 3 > 4

- teraz rezultat będzie 0. Wartość 0 oznacza fałsz.

Przejdźmy z powrotem do uproszczeń instrukcji IF. Szczególnie przydatny będzie wyżej opisany efekt jeśli zmienne będą przyjmować wartości w zależności od porównania. Przez wstawienie mnożników przed porównaniem można całkowicie zrezygnować z konstrukcji IF-THEN.

Użyjmy przykładu: zmienna y powinna przyjmować nową wartość w zależności od x według następującej reguły:

jeśli x większe lub równe 100 to Y ma przyjąć wartość 13; jeśli X jest mniejsze od 100 to Y ma przyjąć wartość 7.

Rezygnując z konstrukcji IF THEN wypróbujmy poniższy program :

Y = -7*(X<100)-13*(X>=100)

Linia ta wygląda skomplikowanie, lecz w zasadzie jest całkiem prosta. Jeśli zmienna X mniejsza od 100, to (X<100) przyjmie wartość -1. Wartość ta pomnożona przez -7 da wynik 7. Tym samym nierówność (X>=100) jest fałszywa i przyjmie wartość 0, w związku z czym 13*0=0. Otrzymamy więc: Y=7-0, czyli Y=7.

Jeśli X będzie większe lub równe 100 to wynik pierwszej nierówności wynosi 0. Nierówność (X>=100) będzie prawdziwa. Jej wynik (-1) zostanie pomnożony przez -13 i otrzymamy 13.

Wykorzystując tą metodę trzeba pamiętać o kilku regułach:

- porównania muszą znajdować się w nawiasach;
- aby otrzymać wynik dodatni mnożnik musi przyjąć wartość ujemną;
- wyniki są pojedyńczymi wyrazami i muszą zostać dodane jeden do drugiego.

OPIS ZESTAWU PUBLIC DOMAIN PACK

AMIGA

nr 16 (kwiecień '92)

Witam! W kwietniowym zestawie znajdziemy dużo użytków, jedną grę logiczną oraz - co staje się już tradycją - kilka ciekawych grafik (jedna digitalizacja, jeden ray-traceing oraz jeden rysunek nieznanego autora).

DiskCruncher - program służy do pakowania całego dysku do jednego lub kilku plików, co czasem pozwala na zmieszczenie danych z dwóch dysków na jednym. Gdy masz na przykład duży zbiór modułów i rzadko je przesłuchujesz możesz użyć DiskCruncher'a aby wygospodarowane miejsce przeznaczyć na nowe moduły itp. Dużą zaletą DiskCruncher'a jest bardzo estetyczne wykonanie interfejsu użytkownika.

IBM-ST-AMIGA pod tą interesującą nazwą (tak naprawdę to interesująca jest tylko ostatnia część) kryje się program, a właściwie zbiór programów pozwalający na BEZPROBLEMOWE korzystanie z dysków IBM'a i ST. Plik skryptowy IBM-ST-AMIGA tworzy nowe urządzenie o nazwie NW0: (odpowiednio NW1:, NW2: itd.) z którego korzystamy tak jak ze zwykłego DF0:. Program świetnie współpracuje z każdym programem w rodzaju File Master, DiskMaster (wszystkie wersje), CLI-mate (to już historia) i inne. Jednak FileMaster v 1.1 jest i tu w czołówce, ponieważ automatycznie rozpoznaje nowe urządzenie i nie trzeba wpisywać ciągle NW0: z klawiatury. Jeżeli nie masz jeszcze File Mastera to szybko zamów poprzedni PDP-Amiga (numer 15). Jest tam File Master i dużo innych ciekawych programów.

Wracając do naszego programiku, ma on niesamowite możliwości, można np. przeglądać IFF'y wprost z dysku ST'owskiego, czy też (o zgrozo!) uruchamiać programy z dysku na którym kolega mający IBM i modem, nagrał np. jakieś download'y! Połączenie NW0: z HiSpeedPascalem (jak wiadomo jest to dokładny odpowiednik Turbo Pascala na Amidze) daje możliwość pracowania z dyskami na których mamy jakieś programy w Turbo, ma się wtedy wrażenie jak gdyby pracowało się na czymś o niebo lepszym od Turbo Pascala (zresztą to nie jest tylko wrażenie) w dodatku z możliwością natychmiastowego zapisania efektów swojej pracy na dysku w formacie IBM!

Aby zainstalować program na swoim dysku należy po pierwsze umieścić plik IBM-ST-AMGA w swoim Startup-Sequence. Teraz należy skopiować podane pliki do następujących katalogów: CD, AMIGADOS, MOUNTME, RUN z katalogu C na dysku PDP-AMIGA do katalogu C na Twoim dysku; z katalogu L: należy skopiować MOUNTLIST oraz NEWHANDLER do tego samego katalogu na nowym dysku. To powinno wystarczyć aby system dobrze pracował.

Jako następny umieściłem program **HamLab v0.91.** Jest to wersja informacyjna ale „91 procent" funkcji działa, a ten program na pewno się przyda. Jest to konwerter graficzny z możliwością obsługi trybu DreamHam. Umożliwia też zapisanie obrazka jako pliku wykonywalnego (co może być bardzo wygodne przy przeglądaniu). Ja umieściłem jednak na PDP IFF'y w normalnej formie, a to tylko dlatego, że niemożliwe jest przekonwertowanie obrazka wykonywalnego z powrotem w zwykły IFF. Program HamLab v0.91 uważam za godny polecenia osobom interesującym się grafiką komputerową.



Kolejny program to **BADv4.13** jest to najnowsza wersja dość znanego programu, służącego do optymalizacji dysków. To co różni tą wersję od poprzednich to bardzo ładna szata graficzna, graficzne przedstawienie dysku, oraz możliwość naprawy dysku. Program ten można zaliczyć do programów naprawdę nowoczesnych - robi swoje, robi to bardzo dobrze i na dodatek jest bardzo wygodny i przyjemny w użytkowaniu.

Ostatnio daje się zauważyć ogromny postęp w wykonaniu programów użytkowych na Amigę. Nowe programy są coraz lepsze, niezawodniejsze, łatwiejsze w obsłudze i mają coraz lepsze interfejsy użytkownika. Niewątpliwie do tego jakby przełomu przyczyniła się reqtools.library, z której korzysta mnóstwo nowych programów - jest to bardzo dobra biblioteka.

Otstatni użytek z kwietniowego Public Domain Packu to **BootXv4.47** z jego wcześniejszymi wersjami mogliście się drodzy Czytelnicy zapoznać, gdyż już gościły one na PDP. Powodem, dla którego zamieszczamy ciągle najnowsze wersje tego programu jest to, że po pierwsze jest to naprawdę jeden z najlepszych (jeśli nie najlepszy) programów antywirusowych, a po drugie programy tego typu wymagają częstych upgrade'ów (powstają ciągle nowe wirusy i stare programy antywirusowe są nieskuteczne).

Był taki okres w historii Amigi, kiedy to wydawało się wszystkim, że problem wirusów został rozwiązany. Było to w czasach panowania wirusów boot-block'owych - wtedy powstał VirusExpert v1.81 autorstwa Paul'a van der Valk'a (poprawiony przez Mac'a). VirusExpert był na owe czasy programem, który nie dawał wirusom boot-block'owym żadnych szans. Wirusy te mają bowiem jedną dużą „wadę" - łatwo je wykryć i usunąć. Właściwie era wirusów boot-block'owych już się skończyła. Niestety zaczęła się era wirusów linkowanych i plikowych, a te są dziesiątki razy bardziej groźne. Jest według mnie jeden powód takiej sytuacji - nie istnieje, żaden ogólny schemat działania takiego wirusa. No cóż, odpowiedzią programistów był BootXv4.47, a także np. VirusExpert v 2.0 stworzony przez Mac'a. Programy te skutecznie dają sobie radę z rozmaitymi wirusami, ale przy założeniu, że trzymamy się jednej podstawowej zasady: należy zawsze mieć najnowszą wersję takiego programu. BootXv4.47 jest programem z kwietnia'92 a więc **super nowość** jak na Polskę. Program ten to potężny kombajn antywirusowy - rozpoznaje 337 boot-block'ów, 170 wirusów boot-block'owych oraz rozpoznaje i zwalcza 40 wirusów plikowych i linkowanych! Na PDP-AMIGA znajduje się PEŁNA wersja programu. Program ten jes FreeWare co oznacza, że można go rozpowszechniać (do czego zachęca sam autor Peter Stuer słowami: „Spread the program not the word!" - świetny gryps).

Ostatnią pozycją kwietniowego Public Domain Pack'u jest **DragonTiles** - ciekawa gra logiczna. Przedstawię teraz w skrócie jej zasady: gracz wybiera sobie plansze do gry, na której ustawione są w różny sposób klocki (do czterech poziomów). Klocków jest 120, na każdym z nich jest jakiś obrazek. Każdy obrazek jest powtórzony na czterech klockach. Zadaniem gracza jest usunięcie wszystkich klocków. Aby usunąć dwa klocki należy je wybrać myszą, ale klocki muszą posiadać ten sam wzór i na dodatek wybrać można tylko takie klocki, które mają po prawej lub lewej stronie wolne miejsce. Nie jest to więc takie proste i bardzo trudno jest usunąć je wszystkie.

Na koniec parę smakowitych **IFF**'ów i... to już wszystko w tym misiącu.

Michał "Amber" Gosztyła

KĄCIK POCZĄTKUJĄCEGO KODERA

CZ.13

Window

AMIGA

W ubiegłym miesiącu omawialiśmy ekran (screen). Dzisiejszy odcinek będzie poświęcony oknom (windows), dzięki którym odbywa się komunikacja użytkownika z programem (poprzez system).

Okno to prostokoątny obszar umieszczony w obrębie ekranu Workbench Screen badź ekranu użytkownika (Custom Screen), które może być tak duże jak duży jest ekran lub mieć tylko jeden punkt szerokości. Możemy dowolnie zmieniać parametry okien: wielkość, położenie na ekranie, widoczność (czyli położenie przed lub za innymi oknami) itd. Wszystkim tym zajmuje się biblioteka intuition. Jeżeli programista będzie modyfikowal struktury dotyczące okien to może to spowodować zawieszenie systemu lub uczyni program niekompatybilny z innymi (wcześniejszymi bądź późniejszymi) wersjami systemu operacyjnego.

Jak już było wspomniane okna są bezpośrednio podłączone do ekranów i dlatego parametry ekranu mają wpływ na parametry okna. Jeżeli mamy ekran w wysokiej rozdzielczości to nie możemy na nim umieścić okna w niskiej rozdzielczości, a także nie możemy umieścić na ekranie okna o innej ilości bitplane'ów niż ma ten ekran. Poruszanie ekranu powoduje przemieszczanie okien na nim umieszczonych gdyż współrzędne okna są podawane względem lewego górnego rogu ekranu.

Normalne okna to prostokątne obszary obwiedzione ramką. Jednak istnieje kilka specjalnych typów okien.

* Backdrop Windows:

Okna typu backdrop będą zawsze za wszystkimi oknami, co oznacza, że możemy dokonywać operacji przemieszczania okien w głąb za pomocą gadżetów głębokości ale i tak będą one ponad oknem typu backdrop. Okna tego typu mają następujące cechy:

- otwierają się zawsze pod wszystkimi otwartymi oknami włączając w to wszystkie otwarte do tej pory inne okna typu backdrop,
- gadżet służący do zamykania okna (Close Gadget) jest jedynym gadżetem systemowym jaki możemy dołączyć do okna typu backdrop. Nie możemy dołączyć gadżetów głębokości służacych do ustawiania okna pod i ponad innymi oknami a także nie możemy dołączyć gadżetu skalowania służacego do zmainy wielkości okna. Wszystkie gadżety użytkownika mogą być oczywiście używane.
- okno typu backdrop zawsze będzie pod wszelkimi innymi oknami.

Okna typu backdrop są bardzo użyteczne gdy chcemy mieć obszar, na którym dokonujemy zmian (na przykład obszar rysowania w programie graficznym) i który jest jednocześnie głównym obszarem. Mozemy teraz używając normalnego okna stworzyć okienko na przykład z narzędziami służącymi do rysowania, które będziemy przesuwać po głównym obszarze bez obawy o ukrycie tego okna pod obszarem głównym bądź zniszczenia obszaru głównego.

* Borderless Windows:

Okna typu borderless to normalne okna za wyjątkiem tego, że są pozbawione otaczającej ramki. Bardzo użyteczna jest kombinacja okna typu borderless z oknem typu backdrop aby pokryć ekran. Dla użytkownika może być bardzo mylące jeżeli okno typu borderless nie pokrywa całego ekranu, gdyż nie będzie się mógl zorientować gdzie są krawędzie takiego okna. Dlatego najlepiej jest kreować okna typu borderless tak duze jak ekran.

* Gimmezerozero Windows:

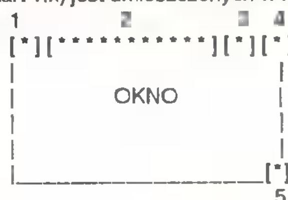
Okna tego typu są jak normalne okna za wyjątkiem tego, iż posiadają dwie przestrzenie rysowania: okno zewnętrzne i okno wewnętrzne. Okno zewnętrzne jest używane do wyświetlania ramki, systemowych gadżetów, etc. podczas gdy okno wewnętrzne jest uzywane tylko przez użytkownika. Jeżeli rysujemy w normalnym oknie to musimy rozpoczynac nasze procedury od pozycji (11,1) (w dół, w prawo) aby nie uszkodzić ramki oraz gadżetów systemowych. Natomiast gdy rysujemy w oknie typu gimmezerozero to nie musimy się martwić o jakieś przesunięcia gdyż pozycja rysowania jest (0,0). Okna typu gimmezerozero mają jednak jedną cechę ujemną a mianowicie zajmują o wiele więcej pamięci i potrzebują o wiele więcej czasu na odswieżanie niż normalne okna.

* Superbitmap Windows:

Jeżeli używamy okna typu superbitmap to musimy przydzielić obszar dla niego samodzielnie. Cechą pozytywną takiego okna jest to, iż możemy zdefiniować o wiele większy obszar niż widzimy w oknie a następnie możemy przesuwać nasz obszar wewnątrz okna. Najlepiej jest połączyć okno typu superbitmap z oknem typu gimmezerozero. Ramka i gadżety nie będą wtedy narysowane w oknie typu superbitmap ale na oknie zewnętrznym okna gimmezerozero. Daje to możliwość pracy z oknem wewnętrznym bez obaw o zniszczenie jakiejś ramki itp.

GADŻETY SYSTEMOWE

Gadżety systemowe pozwalają użytkownikowi na przemieszczanie okna, skalowanie, umieszczanie pod innymi oknami i tym podobne rzeczy bez wpływu naszego programu. Gadżety systemowe są kontrolowane przez Intuition i zawsze wyglądają tak samo. Jedyna rzecz jaką musimy przekazać do Intuition to informację, które gadżety sytemowe potrzebujemy a reszta będzie zrobiona za nas. Pięć gadzetow systemowych (Kickstart 1.x) jest umieszczonych w następujący sposob:



Gadżet 1 - gadżet zamknięcia okna (Close Gadget). Wciśnięcie jego powoduje zamknięcie okna. Jest to jedyny gadżet systemowy, na który musi odpowiedzieć program. Program zostanie poinformowany, że użytkownik wcisnął ten gadżet ale okno musimy zamknąć samodzielnie poprzez wywołanie procedury Intuition.CloseWindow(Window).

Gadżet 2 - gadzet przesuwania okna (Drag Gadget). Trzymanie wciśniętego lewego przycisku myszki na tym gadżecie umożliwia nam przesuwanie okna po ekranie. Jeżeli nasze okno ma nazwę to będzie ona wyświetlana właśnie na tym gadżecie.

Gadżet 3 - gadżet głębokości w dół (Depth-Arrangement Gadget BACK). Wciskając ten gadżet okno zostanie położone pod wszystkimi oknami za wyjątkiem okien typu backdrop.

Gadzet 4 - gadżet głębokości w gorę (Depth-Arrangement Gadget UP) Wciskając ten gadżet okno zostanie położone nad wszystkimi oknami.

Gadżet 5 - gadżet skalowania (Sizing Gadget). Trzymając wciśnięty lewy przycisk myszki na tym gadżecie mamy możliwość skalowania okna.

ODŚWIEŻANIE OKIEN

Podczas gdy okna są przemieszczane po ekranie, jest bardzo prawdopodobne, iż mogą się pokryć (jedno okno zachodzi na inne). Jeżeli użytkownik następnie przesunie okno górne to okno dolne musi być odświeżone. Istnieją dwie metody odświeżania okien:

- Simple Refresh (odświeżanie proste) - Intuition jedynie poinformuje program o tym, że należy odświeżyć okno a program będzie musiał zrobić to samodzielnie.
- Smart Refresh (odświeżanie szybkie) - Intuition przechowuje zawartość przesłonietego okna i wstawią ją automatycznie gdy zaistniała potrzeba odświeżania.

Odświeżanie szybkie potrzebuje więcej pamięci niż odświeżanie proste ale jest od niego szybsze i użytkownik nie musi dbać o to aby odświeżać zawartość okna.

INICJALIZACJA OKNA

Zanim otworzymy okno musimy zainicjować strukturę NewWindow (podobnie jak to miało miejsce przy otwarciu ekranu). Struktura NewWindow wygląda następująco:

struct NewWindow

00 WORD LeftEdge,TopEdge ; lewy i górny róg okna względem lewego górnego rogu ekranu.

04 WORD Width,Height ; szerokość i wysokość okna.

08 UBYTE DetailPen ; rejestr koloru używany do wypisywania tekstu w oknie.

09 UBYTE BlockPen ; rejestr koloru używany do wypełniania obszarów w oknie.

0a ULONG IDCMPFlags ; znaczniki IDCMP (znacznikom IDCMP będzie poświęcony oddzielny artykuł).

0e ULONG Flags ; znaczniki dotyczące okna.

12 APTR FirstGadget ; wskaźnik do pierwszego gadżetu użytkownika (na temat gadżetów będzie poświęcony cały artykuł w Kąciku Początkującego Kodera). Jeżeli ten wskaźnik jest równy zero to oznacza to, iż nie ma żadnych gadżetów podłączonych do tego okna.

16 APTR CheckMark ; wskaźnik struktury Image dla CheckMark czyli znaczka wyboru, używanego przy menu. Na temat struktury Image będzie można przeczytać w kolejnych artykułach Kącika Początkującego Kodera poświęconych strukturom graficznym. Jeżeli wskaźnik jest ustawiony na zero to oznacza to, iż ma być pobrany standardowy zanczek CheckMark.

1a APTR Title ; wskaźnik tekstu zakończonego zerem, który będzie używany jako nazwa okna. Tekst pojawi się na górze okna na pasku przeznaczonym na jego nazwę.

1e APTR Screen ; wskaźnik struktury Screen dla danego ekranu użytkownika na którym to okno ma się pojawić. Jeżeli ten wskaźnik jest równy zero to okno pojawi się na ekranie WorkBench Screen. Jeżeli okno ma się pojawić na ekranie WorkBench Screen to musimy ustawić zmienną Type w strukturze NewWindow na wartość WBENCHSCREEN natomiast gdy ma się pojawić na ekranie użytkownika to zmienna Type musi być ustawiona na wartość CUSTOMSCREEN.

22 APTR BitMap ; jeżeli zamierzamy użyć okna superbitmap to musimy podać Intuition wskaźnik do naszej struktury BitMap (należy pamiętać o ustawieniu znacznika SUPER_BITMAP w zmiennej Flags w strukturze NewWindow oraz można także ustawić zminną GIMMEZEROZERO). Jeżeli okno ma być normalnego typu to wpisujemy tutaj zero.

26 WORD MinWidth,MinHeight ; minimalna szerokość i wysokość okna.

2a WORD MaxWidth,MaxHeight ; maksymalna szerokość i wysokość okna.

2e UWORD Type ; znacznik określający typ okna to znaczy czy to okno jest położone na ekranie WorkBench Screen (wtedy ustawiamy znacznik WBENCHSCREEN), czy też na ekranie użytkownika (wtedy ustawiamy znacznik CUSTOMSCREEN). Przy ustawionym znaczniku CUSTOMSCREEN należy pamiętać aby wpisać adres struktury do pola zmiennej Screen w strukturze NewWindow. Jeżeli mamy podłączony do okna gadżet służący do skalowania tego okna (Sizing Gadget podłączamy za pomocą znacznika WINDOWSIZING (w polu Flags struktur NewWindow) ale o tym za chwilę) to możemy wprowadzić ograniczenia dotyczące skalowania tego okna (pola MinWidth, MinHeight, MaxWidth, MaxHeight określające minimalne i maksymalne rozmiary okna). Jeżeli nie podłączymy tego gadżetu to pola te będą ignorowane. Jeżeli jakaś z wartości jest ustawiona na zero przy inicjowaniu okna to zamiast tej wartości będą użyte wielkości dla okna przy inicjacji to znaczy komórki Width i Height.

Znaczniki dotyczące okna:

* gadżety systemowe:

WINDOWCLOSE - ustawienie tego znacznika informuje Intuition aby dołączył gadżet zamykania okna (Close Gadget) w lewym górnym rogu ekranu. Należy zapamiętać, iż ten gadżet nie zamyka okna a jedynie otrzymamy informację z intuition (w postaci komunikatu IDCMP - o czym będzie traktował następny Kącik Początkującego Kodera) o tym, że użytkownik wcisnął ten gadżet a nasz program musi dokonać zamknięcia okna samodzielnie poprzez wywołanie funkcji Intuition.CloseWindow(Window).

WINDOWDRAG - znacznik ten tworzy cały pasek z nazwą dla okna, który pozwala na przemieszczanie okna po całym ekranie.

WINDOWDEPTH - ustawiamy jeżeli chcemy dać użytkownikowi możliwość umieszczania okna nad i pod innymi oknami. Gadżety głębokości będą przyłączone w prawym górnym rogu okna.

WINDOWSIZING - przyłącza gadżet skalowania w prawym dolnym rogu ekranu umożliwiający użytkownikowi skalowanie okna. (Możemy ustalić wielkości do jakich będzie skalowane okno).

* okna specjalne:

BACKDROP - ustawiamy jeżeli chcemy uzyskać okno typu backdrop.

BORDERLESS - jeżeli ustawimy ten znacznik to otrzymamy okno bez ramki otaczającej.

GIMMEZEROZERO - jeżeli ustawimy ten znacznik to otrzymamy okno typu gimmezerozero.

SUPER_BITMAP - ustawiamy jeżeli chcemy aby nasze okno było typu superbitmap. Jeżeli zamierzamy użyć własnej struktury BitMap przypisanej do tego okna zamiast struktury generowanej przez Intuition to musimy wpisać adres naszej struktury do pola zmiennej BitMap w strukturze NewWindow.

* znaczniki odświeżania

(jeżeli nie używamy okna typu superbitmap to musimy ustawić jeden z tych dwóch znaczników):

SIMPLE_REFRESH - nasz program będzie musiał odświeżać obszar okna samodzielnie. Należy pamiętać, aby na początku odświeżania wywołać procedurę Intuition.BeginRefresh(Window) a na końcu procedurę Intuition.EndRefresh(Window). Procedury te przyśpieszają odświeżanie przez program zawartości okna a jednocześnie czyszczą zmienne systemowe dotyczące tego okna i reorganizują bibliotekę Layers zajmującą się nakładaniem fragmentów graficznych (na przykład okna). Jeżeli otrzymamy komunikat REFRESHWINDOW to musimy jak najszybciej wykonać procedury Intuition.BeginRefresh(Window) i Intuition.EndRefresh(Window) bez względu na to czy chcemy coś odświerzyć w zawartości okna czy też nie.

SMART_REFRESH - gdy ustawimy ten znacznik to Intuition będzie się sama troszczyła o odświeżanie okna i tylko w przypadku gdy użytkownik powiększy okno poprzez skalowanie będziemy musieli tworzyć zawartość okna programowo.

* inne znaczniki:

REPORTMOUSE - ustawiamy gdy chcemy otrzymywać komunikat o położeniu kursora myszki jako współrzędnych x i y (komunikat otrzymywany będzie jako IDCMP - czyli o tym za miesiąc).

NOCAREREFRESH - ustawiamy jeżeli nie chcemy otrzymywać żadnych komunikatów dotyczących odświeżania okna.

RMBTRAP - ustawiamy ten znacznik jeżeli nie chcemy aby użytkownik miał dostęp do menu podczas gdy to okno jest aktywne. Na przykład jeżeli chcemy używać prawego przycisku myszki do innych rzeczy niż wybór opcji z menu.

ACTIVATE - ustawiamy ten znacznik jeżeli chcemy aby okno było aktywne gdy je otworzymy. Użytkownik będzie mógł oczywiście zaktywować inne okno później poprzez wciśnięcie przycisku myszki nad tym oknem (wszystkie pozostałe okna stają się wtedy nieaktywne).

OTWARCIE OKNA

Procedura otwierania okna jest zbliżona do procedury otwierania ekranu Intuition.OpenScreen(NewScreen). Sama idea opiera się na deklaracji struktury NewWindow zbliżonej do struktury NewScreen (patrz Kącik Początkującego Kodera w poprzednim numerze 64 PLUS 4) i zainicjowaniu jej z odpowiednimi zmiennymi. Następnie należy wywołać procedurę Intuition.OpenWindow(NewWindow) z podaną właśnie tą strukturą i jako wynik funkcja ta otworzy okno i zwróci nam wskaźnik struktury Window, który będziemy wykorzystywać do wszelkich operacji z tym oknem. W przypadku niemożności otwarcia okna na skutek złych parametrów lub zbyt małej ilości wolnej pamięci procedura ta zwraca zero. Teraz już nie potrzebujemy struktury NewWindow i możemy ją usunąć lub użyć na przykład do otworzenia kolejnego okna z innymi parametrami. Struktura Window wygląda następująco:

struct Window

00 APTR NextWindow ; wskaźnik struktury Window do kolejnego otwartego okna.

04 WORD LeftEdge,TopEdge ; pozycja lewego górnego rogu okna względem lewego górnego rogu ekranu.

08 WORD Width,Height ; szerokość i wysokość okna.

0c WORD MouseY,MouseX ; pozycja y,x kursora myszki względem lewego górnego rogu okna.

10 WORD MinWidth,MinHeight ; minimalne rozmiary okna

14 WORD MaxWidth,MaxHeight ; maksymalne rozmiary okna

18 ULONG Flags ; znaczniki określające typ okna.

1c APTR MenuStrip ; wskaźnik do struktury Menu określającej wszystkie menu przypisane do tego okna (na temat Menu będzie poświęcony osobny artykuł).

20 APTR Title ; wskaźnik nazwy okna.

24 APTR FirstRequest ; wskaźnik pierwszego z dołączonych requesterów.

28 APTR DMRequest ; wskaźniki do requesterów typu double-menu.

2c WORD ReqCount ; ilość requesterów przypisanych do tego okna.

2e APTR Screen ; wskaźnik struktury Screen, do której to okno jest przypisane czyli na jakim ekranie jest wyświetlane

32 APTR RastPort ; wskaźnik struktury RastPort dla tego okna.

36 BYTE BorderLeft,BorderTop,BorderRight,BorderBottom ; używane gdy mamy okno typu GimmeZeroZero aby określić jak wielkie są ramki okna zewnętrznego: lewa, górna, prawa i dolna.

3A APTR BorderRPort ; RastPort dla ramek przy oknie typu GimmeZeroZero.

3e APTR FirstGadget ; wskaźnik listy gadżetów użytkownika przyłączonych do tego okna (szerzej o gadżetach będziemy mówić w jednym z kolejnych odcinków Kącika Początkującego Kodera). Do tej listy nie są dołączone gadżety systemowe.

42 APTR WindowParent

46 APTR WindowDescendant ; te dwie komórki: WindowParent i WindowDescendant są używane aby odnaleźć poprzednie struktury okien (Window) powiązane z daną strukturą.

4a APTR Pointer ; wskaźnik do danych dla nowego kursora myszki (jeżeli wpiszemy zero to zostanie użyty aktualny kursor).

4e BYTE PtrHeight,PtrWidth ; szerokość i wysokość kursora myszki.

50 BYTE XOffset,YOffset ; przesunięcie punktu głównego kursora myszki względem jego lewego górnego rogu.

52 ULONG IDCMPFlags ; znaczniki IDCMP dla tego okna

56 APTR UserPort ; wskaźnik portu użytkownika służącego do odbierania komunikatów od systemu (komunikaty IDCMP).

5a APTR WindowPort ; wskaźnik portu okna.

5e APTR IntuiMessage ; wskaźnik struktury IntuiMessage dla okna zajmującej się przekazywanie komunikatów IDCMP.

62 BYTE DetailPen, BlockPen ; kolory używane do kreślenia liter i obszarów w tym oknie.

64 APTR CheckMark ; wskaźnik struktury Image zawierającej kształt wskaźnika do menu przełączalnych.

68 APTR ScreenTitle ; wskaźnik nazwy ekranu, jaka będzie wyświetlana gdy okno będzie aktywne.

6c WORD GZZMouseX,GZZMouseY ; pozycja X i Y kursora myszki w odniesieniu do wewnętrznego okna w oknie typu gimmezerozero (względem lewego górnego rogu). Są to porównywalne wartości z komórkami zawierającymi współrzędne kursora myszki.

70 WORD GZZWidth,GZZHEight ; szerokość i wysokość wewnętrznego okna w oknie typu gimmezerozero.

74 LONG ExtData ; nie używane (zarezerwowane na przyszłość).

78 APTR UserData ; dane użytkownika.

7c APTR Layer ; kopia Window.RPort.Layer .

80 APTR IFont ; wskaźnik do czcionki dla tego okna.

Przykład: aby otworzyć okno na ekranie Workbench Screen o wymiarach 100 na 100 i współrzędnych 100,100 z wszelkimi gadżetami systemowymi musimy wykonać poniższą procedurę:

```
Exec equ 4
OldOpenLibrary equ -408
OpenWindow equ -204
CloseWindow equ -72
WBENCHSCREEN equ $0001
SMART_REFRESH equ $00000000
NOCAREREFRESH equ $00020000
WINDOWSIZING equ $0001
WINDOWDRAG equ $0002
WINDOWDEPTH equ $0004
WINDOWCLOSE equ $0008
move.l Exec,a6 ; baza biblioteki Exec
lea IntName(pc),a1
jsr OldOpenLibrary(a6) ; otwarcie biblioteki Intuition
move.l d0,IntBase ; baza biblioteki Intuition
move.l IntBase,a6
lea NewWindow,a0
jsr OpenWindow(a6) ; otwarcie okna
tst.l d0 beq.s Error ; jeżeli w d0 jest zero to oznacza niemożność otwarcia okna
move.l d0,Window
Maus btst #6,$bfe001 ; test lewego przycisku myszy
bne.s Maus ; oczekiwanie na lewy przycisk myszki
move.l IntBase,a6
move.l Window,a0
jsr CloseWindow(a6) ; zamknięcie okna
moveq #0,d0
rts Error
moveq #-1,d0
rts NewWindow dc.w 100,100 ; lewy górny róg okna
dc.w 100,100 ; wymiary okna
dc.b 0,1 ; kolory
dc.l 0 ; znaczniki IDCMP (żaden)
dc.l SMART_REFRESH+NOCAREREFRESH+WINDOWSIZING+WINDOWDRAG+WINDOWCLOSE+WINDOWDEPTH
dc.l 0 ; pierwszy gadżet (nie ma)
dc.l 0 ; CheckMark (standardowy)
dc.l WindowTitle ; nazwa okna
dc.l 0 ; ekran WorkBench Screen
```

```
dc.l 0        ; bez struktury BitMap gdyż nie jest to okno typu SUPER_BITMAP
dc.w 10,10,200,200   ; minimalne i maksymalne wymiary okna
dc.w WBENCHSCREEN
; typ okna
; koniec struktury NewWindow
Window
dc.l 0 IntBase
dc.l 0 WindowTitle
dc.b 'Oto jest okno',0 IntName
dc.b 'intuition.library',0
END
```

I na zakończenie lista procedur używanych przy pracy z oknem:

OpenWindow() - Intuition Window = OpenWindow (NewWindow)
D0 A0

Funkcja otwiera okno na podstawie danych podanych w strukturze NewWindow przekazywanej do tej procedury. Jeżeli zamierzamy otworzyć okno na ekranie WorkBench Screen a jest on zamknięty to zostanie automatycznie otwarty. Jeżeli jednak chcemy otworzyć okno na ekranie użytkownika to musi on być wcześniej otwartyy. Wejście: NewWindow - wskaźnik struktury NewWindow. Wyjście: Window - wskaźnik struktury Window

---

CloseWindow() - Intuition CloseWindow (Window)
A0

Funkcja zamyka okno wcześniej otworzone. Należy pamiętać aby zamknąć wszystkie okna zanim zamkniemy ekran i wszystkie otwarte okna muszą być zamknięte przed wyjściem z programu.

Wejście: Window - wskaźnik struktury Window uzyskany z procedury OpenWindow() - okno musi być ciągle otwarte.

MoveWindow() - Intuition MoveWindow (Window, Delta_X, Delta_Y) A0 D0 D1

Funkcja przesuwa okno na ekranie względem lewego górnego rogu o wartości podane w zmiennych Delta_X i Delta_Y. Delta_X i Delta_Y to przesunięcie o podaną wartość w dół i prawo dla wartości dodatnich oraz w górę i lewo dla wartości ujemnych. Funkcja działa tak samo jak przesunięcie okna za pomocą myszki. Wejście: Window - wskaźnik struktury Window do okna, które zostało wcześniej zainicjowane poprzez wywołanie funkcji OpenWindow() Delta_X - (długie słowo) przesunięcie poziome okna Delta_Y - (długie słowo) przesunięcie pionowe okna.

SizeWindow() - Intuition SizeWindow (Window, Delta_X, Delta_Y) A0 D0 D1

Funkcja skaluje okno podobnie jak to ma miejsce przy pomocy myszki. Skalowanie odbywa się na przesuwaniu prawego dolnego rogu okna względem tego właśnie rogu o wartości podane w zmiennych Delta_X i Delta_Y. Wejście: Window - wskaźnik struktury Window do okna, które zostało wcześniej zainicjowane poprzez wywołanie funkcji OpenWindow(). Delta_X - (długie słowo) ilość pikseli o jaką ma się zmienić poziomy wymiar okna. Delta_Y - (długie słowo) ilość pikseli o jaką ma się zmienić pionowy wymiar okna.

WindowToFront() - Intuition WindowToFront (Window)
A0

Funkcja ustawi okno z przodu, przed innymi oknami. Ma takie samo działanie jak naciśnięcie gadżetu głębokości odpowiadającego za ustawienie okna przed innymi oknami. Wejście: Window - wskaźnik struktury Window do okna, które zostało wcześniej zainicjowane poprzez wywołanie funkcji OpenWindow().

WindowToBack() - Intuition WindowToBack (Window)
A0

Funkcja ustawi okno z tyłu, za innymi oknami. Ma takie samo działanie jak naciśnięcie gadżetu głębokości odpowiadającego za ustawienie okna za innymi oknami. Wejście: Window - wskaźnik struktury Window do okna, które zostało wcześniej zainicjowane poprzez wywołanie funkcji OpenWindow().

SetWindowTitles() - Intuition SetWindowTitles (Window, WindowTitle, ScreenTitle)
A0 A1 A2

Funkcja ma za zadanie ustawić tytuł okna i tytuł ekranu dla aktywnego tego okna po tym jak okno zostało już otwarte. Wejście: Window - wskaźnik struktury Window do okna, które zostało wcześniej zainicjowane poprzez wywołanie funkcji OpenWindow(). WindowTitle - wskaźnik tekstu zakończonego zerem, który będzie ustawiony jako nazwa okna. Jeżeli podamy zero to zostanie wyczyszczony pasek z nazwą okna natomist gdy podamy -1 to zostanie zachowany stary tytuł. ScreenTitle - wskaźnik tekstu zakończonego zerem, który będzie ustawiony jako nazwa ekranu gdy to okno będzie aktywne. Jeżeli podamy zero to zostanie wyczyszczony pasek z nazwą okna natomist gdy podamy -1 to zostanie zachowany stary tytuł.

WindowLimits() - Intuition WindowLimits (Window, MinWidth, MinHeight, MaxWidth, MaxHeight) A0 D0 D1 D2 D3

Procedura zmienia wartości ograniczające skalowanie na wartości podane przy wejściu do tej procedury. Jeżeli jakakolwiek wartość będzie ustawiona na zero to oznacza to, iż wartość ma być niezmieniana. Wejście: Window - wskaźnik struktury Window do okna, które zostało wcześniej zainicjowane poprzez wywołanie funkcji OpenWindow(). MinWidth - (długie słowo) minimalna szerokość okna. MinHeight - (długie słowo) minimalna wysokość okna. MaxWidth - (długie słowo) maksymalna szerokość okna MaxHeight - (długie słowo) maksymalna wysokość okna.

ReportMouse() - Intuition ReportMouse (Window, Wartośç)
A0 D0

Funkcję wywołujemy gdy chcemy otrzymywać komunikaty o współrzędnych kursora myszki. (Będzie to dokładniej wyjaśnione w jednym z następnych artykułów dotyczącym komunikatów IDCMP). Wejście: Window - wskaźnik struktury Window do okna, które zostało wcześniej zainicjowane poprzez wywołanie funkcji OpenWindow(). Wartość - jeżeli podamy wartość ujemną to okno zakończy śledzenie kursora myszki, a jeżeli wartość nieujemną to rozpocznie śledzenie.

BeginRefresh() - Intuition BeginRefresh (Window)
A0

Wywołujemy tę funkcję przed rozpoczęciem odświeżania okna. Funkcja przyśpiesza odświeżanie i tylko te fragmenty okna, które mają być odświeżone będą odświeżane. Wejście: Window - wskaźnik struktury Window do okna, które zostało wcześniej zainicjowane poprzez wywołanie funkcji OpenWindow().

EndRefresh() - Intuition EndRefresh (Window)
A0

Funkcja ta informuje Intuition o tym, że zakończyliśmy odświeżanie okna. UWAGA! Jeżeli otrzymamy od Intuition komunikat REFRESHWINDOW mówiący o tym, że należy odświeżyć okno to powinniśmy jak najszybciej wywołać procedury BeginRefresh() i EndRefresh nawet jeżeli nie będziemy nic odświeżać programowo. Wejście: Window - wskaźnik struktury Window do okna, które zostało wcześniej zainicjowane poprzez wywołanie funkcji OpenWindow().

W tym miesiącu to już wszystko na temat okien.

W następnym Kąciku Początkującego Kodera będziecie mogli przeczytać o znacznikach IDCMP.

Marcin "Duddie" Dudar

ZWARTY KOD

Nie tylko w asemblerze można pisać zwarte programy, lecz również w C. Jeśli zastąpi się w programach funkcję **printf()** przez **puts()**, to zaoszczędzimy kilka bajtów.

Print() pozwala na znaczną elastyczność postaci wyprowadzanego tekstu, lecz kod tej funkcji jest obszerniejszy niż przy **puts()**. Dla tego, kto chce tylko „wydawać" teksty wystarczy w zupełności funkcja **puts()**.

Zastępując tradycyjne **main()** w kodzie źródłowym przez **__main** oszczędza się miejsce. Linker zrezygnuje w tym wypadku z dużej części Starup-Codes. Przy tym można zauważyć, że Amiga nie otwiera więcej standardowych kanałów wejścia-wyjścia. Znaczy to, że wejście i wyjście tekstu przez okna CLI jak również przełączenie na inne urządzenie nie będzie więcej wspierane.

Wszystkie funkcje C, które korzystają z tego kanału, nie mogą być więcej użyte. Przede wszystkim dotyczy to: **printf()**, **puts()** i **getchar()**.

Kontynuowane będzie w dalszym ciągu przesyłanie parametrów z CL, choć zmienia się się ich wykorzystanie w programach C. Wprowadzane linie nie będą już więcej rozbite na pojedyńcze parametry, programista musi w razie potrzeby samemu dokończyć to zadanie. Zmienna **argc** będzie już tylko imitacją, a ***argv** nie będzie więcej żadnym polem (ARRAY) lecz jedynie strzałką w łańcuchu.

Przedstawiamy dwa programy w C, które wykonują takie same zadania, lecz wykazują znaczne różnice w długości. Oba służą do wydawania przekazywanych parametrów w oknie CLI. Pierwszy program został napisany konwencjonalnie i jego długość wynosi 4959 bajtów.

Wywołanie pierwszego programu do kompilacji brzmi:

```
cc Program
ln Program. o -lc
```

Drugi został napisany jako wersja oszczędnościowa programu pierwszego. Przede wszystkim zwróćcie uwagę na zmiany przy przekazywaniu parametrów CLI. Dla wydawania tekstu w oknach CLI program ten używa własnych funkcji **put()**, które otwierają standardowy kanał wyjścia dos. library przez **open()**, wydają łańcuch i zamykają znów kanał funkcją **close()**. Alternatywą to tego jest funkcja **output()**. Chociaż kod źródłowy tej wersji programu jest trochę dłuższy od pierwszej, to kod wynikowy jest krótszy o 428 bajtów. Dla kompilowania niezbędne będą następujące instrukcje :

```
cc Program +1
ln Programm.o -m -lc32
```

Kto - obok C - zna również asembler może zaoszczędzić jeszcze kilka bajtów jeśli poprawi wytworzone z kompilatora źródło. Ale to już jest praca dla profesjonalistów.

```
#include <exec/types.h>
#include <functions .h>

main(argc,argv)
int argc;
char *argv[];
{
printf("Argument:%sn",argv[1]);
}
```

Krótki kod źródłowy - długi program. Po skompilowaniu programu kompilatorem AZTEC-C otrzymamy dane długości 4956 bajtów.

```
#include <exec/types.h>
#include <libraries/dos.h>
#include <functions.h>

put (str)
void *str
{
void *fhd;
fhd=Open("*",MODE_OLDFILE);
Write(fhd,str,strlen(str));
Close(fhd);
}
_main(argc,argv)
int argc;
char *argv;/* bez [] */
}
argv[strlen(argv)-1]=0;
put("Argument:");
put(argv);
put("n");
}
```

Kod źródłowy tego programu jest wprawdzie dłuższy, lecz skompilowany program jest krótszy o 428 bajtów.

opr. RG

Zmienne łańcuchowe

Chcąc używać w Amiga-Basic zmiennych nadajemy im po prostu nazwę. W języku C natomiast musimy wszystkie zmienne zdefiniować przed ich wstawieniem. Podobie jak w Basic'u również w C można znaleźć zmienne różnych typów, które są rozróżniane przez system. Dlatego zawsze należy - deklarując zmienne - podać ich typ, dzięki czemu kompilator będzie wiedział ile, dla każdej z nich, przeznaczyć miejsca w pamięci.

Przy deklaracji można przydzielać zmiennym wartość startową. Metoda ta zaoszczędzi nie tylko pracy, lecz również pamięci w komputerze. Program będzie krótszy jeśli zainicjuje się zmienne już przy deklaracji, a nie dopiero w programie.

Przyklad:

deklaracja liczby calkowitej (integer):

```
int Variable = 123 ;
```

Metoda ta jest szczególnie przyjemna przy wykorzystywaniu łańcuchów znaków. Ponieważ język C nie zna żadnych zmiennych typu STRING, tekst trzeba zlożyć ze zmiennych typu CHAR w tablicy (ARRAY). Taka deklaracja eliminuje podawanie wielkości pola:

```
char String [] = "hallo";
```

Również razem zestawione typy danych, jak struktura, mogą zostać zainicjowane przy deklaracji. Zostaną rozdzielone przy tym poszczególne wartości przez przecinki i opisane klamrami. Przy końcu deklaracji koniecznie musi znajdować się średnik. Nawet przy wielowymiarowych tablicach jest możliwa inicjacja, przykładowo można w taki sposób zrealizować STRING ARRAY, że różne numery błędów zawierać będą błędne komunikaty:

```
char StringArray [2] [14] =
{
  { "brak pamięci" },
  { "dysk full" }
} ;
```

Ogólna inicjacja będzie objęta przy tablicach wielowymiarowych klamrami. Poszczególe wymiary znajdą się w klamrach rozdzielone przecinkami. Wadą wielowymiarowych łańcuchów znakowych jest brak automatycznego obliczania ich długości przez kompilator. Należy podać długość STRING i doliczyć znaki, gdyż do każdego STRING kompilator C dodaje jedno zero.

opr. RG